EXHIBIT A

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

IMPLICIT NETWORKS, INC.	§	
	§	
Plaintiff,	§	
	§	
	§	
	§	
	§	
vs.	§	**Case No. C 10-4234 SI**
	§	
	§	
	§	
JUNIPER NETWORKS, INC	§	
	§	
Defendant.	§	

REBUTTAL EXPERT REPORT OF PETER ALEXANDER, PH.D.

REGARDING NON-INFRINGEMENT OF

U.S. PATENTS NO. 6,629,163 AND 7,711,857

September 11, 2012

TABLE OF CONTENTS

1 INTRODUCTION AND BACKGROUND

1. Plaintiff Implicit Networks, Inc. ("Implicit") alleges that defendant Juniper Networks, Inc. ("Juniper") infringes United States Patent No. 6,629,163 ("the '163 patent") entitled "Method and System for Demultiplexing a First Sequence of Packet Components to Identify Specific Components Wherein Subsequent Components are Processed Without Re-Identifying Components" and United States Patent No. 7,711,857 ("the '857 patent") entitled "Method and System for Data Demultiplexing" (collectively "Patents-in-Suit"). The '857 patent is a continuation of the '163 patent.

2. I have been retained as an independent expert witness in this case to provide opinions and conclusions as to whether Juniper has infringed the asserted claims of the patents-in-suit. In particular, I have been asked to offer a rebuttal to the Aug. 10, 2012 Expert Infringement Report of Dr. Nettles (herein "Nettles Report").

3. I submit this expert report in compliance with Rule 26(a)(2) of the Federal Rules of Civil Procedure on issues pertaining to the technologies of the patents-in-suit, and in particular regarding Juniper's products and my opinions that Juniper does not infringe the Patents-in-Suit, as set forth in this expert report and in any supplemental reports or declarations that I may prepare for this litigation in the future.

4. As explained in further detail below, my opinion is that the Juniper accused products do not infringe any of the asserted claims of the '163 patent or '857 patent and that Dr. Nettles has failed to satisfy Implicit's burden of proving any infringement by Juniper.

1.1 Asserted Claims & Accused Products

5. I understand that Implicit has alleged that Juniper's systems infringe claims 1, 15 and 35 of the '163 patent and claims 1, 4 and 10 of the '857 patent (collectively "asserted claims").

6. I further understand that in this case Plaintiff has accused fourteen different products: (1) SRX100 Services Gateway; (2) SRX210 Services Gateway; (3) SRX220 Services Gateway; (4) SRX240 Services Gateway; (5) SRX650 Services Gateway; (6) SRX1400 Services Gateway; (7) SRX3400 Services Gateway; (8) SRX3600 Services Gateway; (9) SRX5600 Services Gateway; (10) SRX5800 Services Gateway; (11) J2320 Services Router; (12) J2350 Services Router; (13) J4350 Services Router; (14) J6340 Services Router. I sometimes refer to these products below as the "accused products" or simply the "Juniper products." I also understand that Implicit previously accused a number of other products and combinations of products of infringing the patents-in-suit, but that Implicit has now withdrawn its infringement claims for all products other than those specifically listed in this paragraph.

7. I understand that the Court issued a Claim Construction Order on February 29, 2012. ("Markman Order"). The opinions in this report depend upon and are expressly based on the Court's Markman Order and my review of the accused products.

8. I have reviewed the patents-in-suit, the prosecution histories for the '163 patent and the '857 patent (including reexamination proceedings), technical documents regarding the accused products (including source code for the accused products), Implicit's infringement contentions, the Markman Order and other court papers, Dr. Nettles's Report and the materials he reviewed in informing his opinions, deposition testimony, written discovery responses, and other relevant documents in forming the opinions expressed in this report. My opinion is based on my review of such materials, together with my education, training, and experience in the relevant field. A list of materials I considered is attached as Attachment A.

9. If Dr. Nettles or Implicit offers any new opinions regarding infringement or other topics relating to the subject matter of his August 10 report, I reserve the right to supplement my report and provide rebuttal opinions.

1.2 Implicit's New Infringement Theories

10. I note that a number of the infringement positions and theories that are currently included in Dr. Nettles's expert report do not appear to have been disclosed in Implicit's infringement contentions of November 2011 (which I understand was the deadline for Implicit to disclose its infringement positions and theories).

11. For example, Implicit's Nov. 2011 infringement contentions pointed solely to an "application identification caching" feature in Juniper accused products to establish the "performing the processing of the identified component with the packet and the retrieved state information" limitation. Dr. Nettles's Report, however, cites to source code that is entirely unrelated to the "application identification caching" feature Implicit identified. As another example, Dr. Nettles's report cites to code relating to "captive portal content delivery," "session interest," and "session ignore" functions that allegedly satisfy certain elements of the asserted claims. No such allegations are contained in Implicit's Nov. 2011 infringement contentions (which do not even mention "captive portal content delivery," and mention only in passing "session interest" and "session ignore" in its introductory graphics among dozens of other features (and do not claim they satisfy any particular element of the asserted claims). As another example, the Nettles Report at ¶87-100 provides new theories of indirect infringement based on alleged inducement by Juniper.

12. I understand that Juniper has taken the position that Implicit's attempt to belatedly add new infringement theories to the case are improper and prejudicial to Juniper. Accordingly, nothing in my report or analysis below should be taken as a concession that Implicit can or should be

permitted to proceed under these new theories during the summary judgment phase of this case or trial.

13. I further understand that the Court ordered Implicit to provide pinpoint citations to all source code supporting its claims of infringement as set forth in Implicit's Nov. 2011 infringement contentions. The Court further ordered Implicit to provide these citations on an element-by-element basis. I note that not all of the contentions set forth in Implicit's subsequent infringement contentions (served in June 2012) comply with this requirement. Again, my understanding is that Juniper is expressly reserving its objections to Implicit's failure to comply with these Court orders, notwithstanding my analysis below.

2 QUALIFICATIONS

14. My academic credentials include a Ph.D. from the Massachusetts Institute of Technology in Electrical Engineering, a Masters degree from the University of Illinois (Urbana) in Electrical Engineering, and a Bachelor of Science in Electrical Engineering from the University Canterbury, New Zealand. My Curriculum Vitae, including a list of all cases in which I have testified during the past four years, is attached as Attachment B. A list of case in which I have testified in deposition or trial is provided as Attachment C.

15. The patents-in-suit claim purported inventions involving in the field of computer networking software. My professional experience in this area is extensive. For example, I have multiple years of software design and development experience involving networking products such as routers, bridges and gateways. From 1989 through 1992, in my capacity as the leader of a software development group at Fibronics International, Inc., I was responsible for the creation of source code used in products that provided Internet layer services to other application-level software products. Specifically, I was responsible for the development of core Internet software

protocols, such as Transport Control Protocol ("TCP") and Internet Protocol ("IP"), as well as higher level computer applications such as Domain Name Services ("DNS"), File Transfer Protocol ("FTP"), Network File System ("NFS"), and Telnet services.

16. The IP software included functional components such as Address Request Protocol ("ARP"), Reverse Address Resolution Protocol ("RARP"), and Internet Control Message Protocol ("ICMP").

17. In addition I was involved with the design of software for Local Area Network ("LAN") bridges that carried diverse, multi-protocol traffic such as Appletalk and Novell IPX, so I also have a broader understanding of other network protocol families.

18. Subsequent to this direct involvement with the design of software for bridges, routers and servers, I have gained substantial experience with network engineering for corporate local area networks ("LANs") and Wide Area Networks ("WANs"). For example, from 1997 to 2003, I was responsible for the implementation of LAN and WAN networks configured with commercial products such as firewalls, load balancers, web servers, application servers, routers, layer 2 switches, and client-server applications generally.

19. During this period I was typically responsible for corporate networks and I managed Information Technology ("IT") staffs of around 10-20 people. The specific companies involved were Syntricity, Inc., InfrastructureWorld.com, CareerPath.com and Platinum Software Corporation. My experience includes network configurations for file servers, email servers, firewalls, routers, layer 2 switches, Java application servers and web servers.

20. As a result of my involvement with the design of these networked components I am also well versed in the computer source code for software systems such as the Linux operating

system, Apache web server, and the Tomcat application server that interact with the underlying network services.

21. I have over 30 years of software design and development experience using various programming languages appropriate for networking applications and infrastructure. These languages include C, C++, Java, as well as assembly language for various processors. As discussed above, I have developed TCP/IP protocol source code and related TCP/IP applications code using the C programming language. The TCP/IP source code developed was used as an embedded protocol stack in routers and bridges, and was also used as the core TCP/IP communications capability by companies marketing mainframe computers. During the early part of my career during the 1970s and 1980s, I was responsible for many defense-related software implementations that required embedded applications designed for communications and signal processing. These systems were developed in native assembly language or compiled C code to achieve high efficiency.

22. Further details on my background and experience, including a list of cases in which I have testified as an expert at trial or at deposition is contained in my curriculum vitae, which is attached as **Attachment B.** As shown in Attachment B, I have provided consulting services for many companies over the years that required me to perform in-depth analysis and review of software code and highly complex software systems.

3 METHODOLOGY USED

23. I have been informed of the following general principles of patent law and have applied these principles based upon my understanding of them in formulating my opinions as to the issues of non-infringement of the asserted claims.

24. I am informed that patent infringement under 35 U.S.C. 271(a) consists of making, using, offering to sell, or selling a patented invention within the United States, or importing a patented invention into the United States, without authorization. Determining whether there is infringement of a patent involves two steps. First, each asserted claim must be construed to determine its proper scope and meaning to one of ordinary skill in the art. Second, each element (or limitation) of the asserted claims, as construed, is compared to the accused device or process to determine whether the element (or limitation) is found in the accused device or process.

25. I understand that claim construction is an issue of law, which the Court decides by interpreting claim terms as they would have been understood by one of ordinary skill in the relevant art at the time of the invention. I understand that the claims of a patent are interpreted in light of the specification – that is, the claim language itself, the written description, and the figures in the patent – as well as the patent's prosecution history. I further understand that I am to apply the Court's claim constructions set forth in the Markman Order, and I have done so in this report. I also understand that terms that the Court has not construed are to be given their ordinary and customary meaning to one of ordinary skill in the art at the time of invention, absent any teaching of a different meaning within the specification or prosecution history. As a result of my education and experience, I understand how the asserted claims of the patents-in-suit would be understood by a person of ordinary skill in the art in the relevant timeframe.

26. I understand that Implicit, as the party asserting infringement, has the burden of proving by a preponderance of the evidence that each accused product practices all the elements of at least one of the asserted claims of the patents-in-suit. I understand that a "preponderance of the evidence" is such evidence that causes a trier of fact to be persuaded that the fact sought to be

proved is more likely than not to be true. I have applied the "preponderance of the evidence" standard throughout my report.

27. I also understand that is it not Juniper's burden to prove non-infringement. Accordingly, in this report, I have identified and addressed several limitations of the asserted claims that in my opinion are not present in or practiced by the accused products and/or Juniper. Although I believe that I have discussed each and every limitation of the asserted claims below, any omission should not be taken to mean that I agree with Dr. Nettles's opinion that any claim limitation is present in or practiced by the accused products and/or Juniper.

28. To find infringement by an accused system or method, the patentee must show the presence of every element of a given claim (literal infringement) or its equivalent (infringement under the doctrine of equivalents) in the accused system or method. Dr. Nettles's report does not address the doctrine of equivalents, nor do Implicit's infringement contentions. Thus, I will not address this doctrine further.

29. I further understand that an accused system or method literally infringes a patent claim if all the elements of the claim, as properly construed by the Court, are present in the accused system or method. If any element of the claim does not appear in the accused system or method exactly, that claim is not literally infringed.

30. I also understand that a method claim cannot be infringed unless each step of the method claim is actually performed. The fact that equipment might be capable of performing the steps of a method claim is insufficient to establish infringement of the method claim.

3.1 Direct Infringement

31. I understand that each construed claim must be compared to the accused device or process to determine whether every claim element is found in the accused device or process. A

patent is directly infringed only when the accused product contains each and every claim limitation, or where a single party performs each and every step of a method claim. If any claim limitation is absent from the accused device or method, there is no literal infringement as a matter of law.

32. I also understand that a method claim cannot be directly infringed unless each element of the method claim is actually performed by a single actor. The fact that equipment or software might be capable of performing the elements of a method claim is insufficient to establish infringement of the method claim. Similarly, an apparatus claim cannot be shown to be infringed through the mere identification of a hypothetical configuration satisfying the elements of the claim.

33. I further understand that where the actions of multiple parties combine to perform every step of a claimed method, the claim is directly infringed only if one party exercises control or direction over the entire process such that every step is attributable to the controlling party.

3.2 Indirect Infringement

34. I understand that a patent may be infringed indirectly. I understand that two types of indirect infringement are induced infringement and contributory infringement. I further understand that indirect infringement requires a threshold showing of direct infringement.

3.2.1 Inducement of Infringement

35. I understand that induced infringement requires knowledge that the induced acts constitute patent infringement. I understand that a party induces patent infringement when it purposefully causes, urges, or encourages another to infringe a patent. The requirements for induced patent infringement include: the alleged infringer actively encouraged or instructed another person on how to use a product or perform a process in a way that infringes the patent claims; the alleged infringer knew of the patent (or if the alleged infringer was willfully blind of the patent); the alleged infringer knew or should have known that the encouragement or instructions would induce

infringement of the patent; and the patent was in fact directly infringed. More specifically, I understand that for a finding of inducement the alleged infringer must have had a specific intent to induce others to directly infringe and must have knowingly encouraged the infringement, not merely caused the acts that constitute direct infringement.

36. Further, I understand that inducing infringement requires some affirmative act to bring about the desired result (in this case, infringement). It is not enough to demonstrate inducement that a party failed to take steps to prevent others, including its affiliates, from committing acts of direct infringement. Inducement cannot be predicated on an omission; mere inaction does not suffice to establish inducement. I also understand that a finding of inducement further requires a showing that the alleged inducer affirmatively encouraged the performance of every limitation of the claim at issue. Encouragement of inducement of less than all of the claim limitations does not constitute inducement of infringement. The Plaintiff must provide sufficient evidence to demonstrate that Defendant had specific intent to cause the acts which constitute the infringement.

3.2.2 Contributory Infringement

37. I further understand that a party that makes, uses, or sells a material component of a patented invention which is not itself a staple article of commerce and with knowledge that the component is especially made for use in an infringement of a patented invention can be liable for contributory infringement even though that party does not directly infringe. There is no contributory infringement if the component that is made, used or sold is a staple article of commerce that is not especially adapted for infringement, nor if the party that makes uses or sells the component has no knowledge that the component was so adapted. A party cannot be liable for contributory infringement if the article in question is suitable for substantial non-infringement use.

4 THE PATENTS-IN-SUIT

38. If asked at trial, I may provide a tutorial of the '163 patent and the '857 patent, as well as the significance and meaning of these patents in the context of Implicit's allegations and the relevant technological background.

4.1 Background to Patents-In-Suit

39. The '163 patent, entitled "Method and system for demultiplexing a first sequence of packet components to identify specific components wherein subsequent components are processed without re-identifying components," issued on Sept. 30, 2003 to Balassanian. The application that resulted in the '163 patent, Application No. 09/474,664, was filed on December 29, 1999.

40. The '857 patent, entitled "Method and system for data demultiplexing" issued on May 4, 2010 to Balassanian. The application that led to the '857 patent, Application No. 11/933,022 was filed on October 31, 2007. The application was a continuation of Application No. 10/636,314 (filed August 26, 2003), which is a continuation of Application No. 09/474,664, filed on December 29, 1999, now Patent No. 6,629,163.

4.2 Asserted Claims of the '163 & 857 Patents

4.2.1 '163 Patent

4.2.1.1 Claim 1 – '163 Patent

1. A method in a computer system for processing a message having a sequence of packets, the method comprising:
providing a plurality of components, each component being a software routine for converting data with an input format into data with an output format;
for the first packet of the message,
dynamically identifying a non-predefined sequence of components for processing the packets of the message
such that the output format of the components of the non-predefined sequence match the input format of the next component in the non-predefined sequence,
wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received; and
storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message; and

for each of a plurality of packets of the message in sequence,
for each of a plurality of components in the identified non-predefined sequence,
retrieving state information relating to performing the processing of the component with the previous packet of the message;
performing the processing of the identified component with the packet and the retrieved state information; and
storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.

4.2.1.2 Claim 15 – '163 Patent

A method in a computer system for demultiplexing packets of messages,
the method comprising: dynamically identifying a non-predefined sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components,
wherein different non-predefined sequences of components can be identified for different messages, each component being a software routine,
and wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components; and
for each packet of each message, performing the processing of the identified non-predefined sequence of components of the message
wherein state information generated by performing the processing of a component for a packet is available to the component when the component processes the next packet of the message.

4.2.1.3 Claim 35 – '163 Patent

A computer-readable medium containing instructions for demultiplexing packets of messages,
35a. by method comprising: dynamically identifying a message-specific non-predefined sequence of components for processing the packets of each message
35b. upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific non-predefined sequence identified when the first packet was received,
35c. and wherein dynamically identifying includes selecting individual components to create the message-specific non-predefined sequence of components;
35d. and for each packet of the message, invoking the identified non-predefined sequence of components in sequence to perform the processing of each component for the packet
35e. wherein each component saves message-specific state information so that that component can use the saved message-specific state information when that component performs its processing on the next packet of the message

4.2.2 '857 Patent

4.2.2.1 Claim 1 – '857 Patent

1. Preamble. A method in a computer system for processing packets of a message, the method comprising:
receiving a packet of the message and a data type of the message;

analyzing the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence, wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;
storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;
for each of a plurality of components in the identified sequence:
performing the processing of each packet by the identified component;
and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.

4.2.2.2 Claim 4 – '857 Patent

A method in a computer system for processing a message, the message having a plurality of headers, the method comprising:
analyzing the plurality of headers of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence,
wherein analyzing the plurality of headers of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;
storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;
for each of a plurality of components in the identified sequence:
performing the processing of each packet by the identified component; and
storing state information relating to the processing of the component with the packet for use when processing the next packet of the message.

4.2.2.3 Claim 10 – '857 Patent

A computer readable storage medium, other than a data transmission medium, containing instructions for processing packets of a message, the instructions comprising at least one computer executable module configured to:
receive a packet of the message and a data type of the message;
analyze the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence, wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received;
store an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message;
for each of a plurality of components in the identified sequence: perform the processing of each packet by the identified component; and store state information relating to the processing of the component with the packet for use when processing the next packet of the message.

4.3 File History

4.3.1 Prosecution History

4.3.1.1 Original Prosecution Of The '163 Patent

41. During the original prosecution of the '163 patent, the patentee's initially proposed claims were rejected by the PTO as anticipated by at least three patents, namely U.S. Patent No. 5,870,479 to Feiken et al. ("Feiken"), U.S. Patent No. 5,425,029 to Hluchyj et al. ("Hluchyj"), and U.S. Patent No. 5,568,478 to Van Loo, Jr. et al. ("Van Loo").

42. In response to the PTO's rejection, the patentee cancelled the claims and added a new set of claims that added a "storing" step that recited a sequence of components that did not need to be re-identified for packets arriving after the first packet. The patent subsequently issued on September 30, 2003.

4.3.1.2 Original Prosecution Of The '857 Patent

43. During the original prosecution of the '857 patent, the patentee's initially proposed claims were rejected by the PTO over a single prior art reference: U.S. Patent No. 6,785,730 to Taylor ("Taylor"). 6/24/2009 Initial Office Action at 3-9.

44. In response to the PTO's rejection, the patentee amended the claims to add several new limitations in an attempt to distinguish the prior art from Taylor. 9/24/2009 Amendment at 4-8. The patentee also made several arguments as to why the new limitations distinguished the claims from Taylor. *Id.* at 11-13.

45. Because of the similarity between '163 patent and the '857 patent, the Examiner indicated that the claims containing the newly added limitations would be allowable if a terminal disclaimer was filed to overcome the obviousness-type double patent rejection. 12/11/2009 Final Rejection at 3. The patentee filed a terminal disclaimer and three independent claims were allowed.

3/10/2010 Notice of Allowance at 1. Each of these three independent claims included the "dynamically identifying" and "selecting individual components" limitations added during the '163 reexamination. Moreover, each claim expressly required that "dynamically identifying the sequence of components" must occur "after the first packet of the message is received."

4.3.1.3 First Re-Examination Of The '163 Patent

46. On January 17, 2009, the PTO granted a request for *ex parte* reexamination of the '163 patent, finding that a substantial new question of patentability existed. The *ex parte* reexamination was granted in view of a dissertation submitted by David Mosberger to the Department of Computer Science at The University of Arizona entitled "Scout: A Path-Based Operating System" (Mosberger"). The found a substantial new question of patentability existed as to all the claims of the '163 patent and rejected all of the claims over Mosberger.

47. Implicit attempted to distinguish Mosberger by alleging that the system of the '163 patent "configured paths at run-time (i.e., after the first packet is received)," while the system of Mosberger "configures paths (formed from a sequence of components) before receiving the first packet of the message." 9/01/2009 Amendment at 11. Indeed, Implicit alleged during the prosecution of the '163 patent that its claims required that the sequence of components be "created dynamically":

> In other words, the '163 Patent clearly states that the invention requires the sequence of conversion routines (that form the paths) to be identified at run-time, and disavows prior art systems (like Mosberger) that use pre-configured paths, which are defined at "build-time" before the first packet of a message is received.
> 9/01/2009 Amendment at 18.

48. The PTO responded by arguing (among other things) that the distinction upon which Implicit relied was not actually included in the claim language of the '163 patent. 12/04/2009 Final Office Action at 13-14 (claimed invention "not recited as being dynamic in nature"). In response,

Implicit amended the claims to add the "dynamically" language to the claims, as well as the phrases "selecting individual components" and "after the first packet is received." 12/18/2009 Response to Final Rejection at 10. The PTO initially refused to enter the amendments. 1/21/2010 Advisory Action. A subsequent Examiner interview was conducted, and Implicit submitted additional amendments adding the "non-predefined" limitation a few days later. 2/8/2010 Amendment After Final. The Examiner allowed the claims in light of these amendments and its decision expressly relied on Implicit's argument "that Mosberger does not dynamically identify sequences of components . . . ." 3/2/2010 Notice of Intent to Issue Ex Parte Reexamination Certificate at 4.

4.3.1.4 Second Re-Examination History of the '163 Patent

49. On April 3, 2012, the PTO granted Juniper's request for the *inter partes* reexamination of the '163 patent and issued an initial office action rejecting all the asserted claims of the '163 patent as anticipated or obvious over two prior art references: (1) Decasper, D., et al., "Router Plugins: A Software Architecture for next Generation Routers," Computer Communication Review, a publication of ACM SIGCOMM, Vol. 28, No. 4, Oct. 1998 ("Decasper98"); and (2) U.S. Patent 6,243,667 to Kerr et al., issued Jun. 5, 2001 ("Kerr").

50. In order to distinguish the prior art cited by the Examiner, Implicit has argued (among other things) that (1) "selecting individual components" requires the system to "ensure that the input format of one component matches the output format of the previous component" by "adding functionality to check or compare the output format. . .with the input format"; (2) an "IP router and its components, by definition" do not perform conversion because they handle packets in a "single format," "IP (Internet Protocol) format"; (3) the prior art does not perform the step of "storing state information relating to the processing of the component with packet for use when processing the next packet of the message" because "IP routers simply do not contemplate this type of state

management"; and (4) the sequence of components is "predefined" or "predetermined" before the "first packet of a new flow." *See, e.g.*, 6/4/2012 Implicit's Response to '163 Reexam at 19-22.

4.3.1.5 First Re-Examination History of the '857 Patent

51. On May 10, 2012 the PTO granted Juniper's request for the *inter partes* reexamination of the '857 patent and an initial office action rejecting all the asserted claims of the '857 patent as anticipated and/or obvious in view of the following references and combinations: (1) Decapser98; (2) Decapser98 in view of IBM Local Area Network Concepts and Products: Routers and Gateways (May 1996) ("IBM96"); (3) Decasper98 in view of IBM96 and Nelson, M. and J-L Gailly, The Data Compression Book, M&T Books (2nd ed. 1996) ("Nelson"); and (4) Kerr in view of Pfeifer, T., and R. Popescu-Zeletin, "Generic Conversion of Communication Media for Supporting Personal Mobility," Lecture Notes in Computer Science Multimedia Telecommunications and Applications, Third Int'l COST 237 Workshop Nov. 25-27, 1996, Proceedings, Springer, pp. 104-129 (G. Ventre, et al. eds. 1996) ("Pfeifer96").

52. In order to distinguish the prior art in cited by the Examiner, Implicit has reemphasized the same arguments made in its response to the Office Action in *inter partes* reexamination of the '163 patent, among others. *See* 7/10/2012 Implicit's Response to '857 Reexam.

4.4 Construed Claim Terms

53. It is my understanding that in providing opinions regarding whether or not the accused software infringes the asserted claims of the patents-in-suit I should apply the claim constructions of claim terms provided by the Court. The following constructions were determined by the Court in the Feb. 29, 2012 Markman Order.

Claim Term	Court Constructions
"non-predefined sequence of components" '163 patent, Claims 1, 15, 35	"a sequence of software routines that was not identified before the first packet of a message was received."

"…processing" and "all variants" '163 patent Claims 1, 15, 35 '857 patent Claims 1, 4, 10 [Variants include: "Processing [the] [packets of] [a/each] message," "Processing [the/a plurality of] packets of [a/the/each] message[s]," "Process[ing][es] the next packet of the message," et]	"manipulating data with a program."
"Input/Output Format." '163 patent Claim 1 '857 patent Claims 1, 4, 10	"structure or appearance of data to be processed" and the "structure or appearance of the data that results from processing."
"Selecting individual components" '163 patent Claims 1, 15, 35 '857 patent claims 1, 4, 10	"selecting the individual software routines of the sequence so that the input and output formats of the software routines are compatible."
"based on the first packet of the message" '163 patent Claim 15	"relying on information in the first packet of the message."
"Message[s]" '163 patent Claims 1, 15, 35 '857 patent Claims 1, 4, 10	"a collection of data that is related in some way, such as a stream of video or audio data or an email message."
"State information" '163 patent claims 1, 15, 35 '857 patent Claims 1, 4, 10	"information specific to a software routine for a specific message that is not information related to an overall path."

4.4.1 "selecting individual components"

54. The Court went on to provide additional guidance regarding the meaning of "selecting individual components" in its Markman Order:

> The Court finds that, based on the claim language, teachings of the specification and the prosecution history, a necessary part of **selecting individual components** is determining the compatibility between the output of one software routine and the input of the next Therefore, **selecting individual components** is construed as "selecting the individual software routines of the sequence so that the input and output formats of the software routines are compatible." Markman Order at 11:6-10. (Emphasis in original.)

55. I understand based on statements made by Implicit (see, e.g., Balassanian Depo. Tr. dated August 16-17, 2012) that Implicit agrees with the Court's statement that this active "determining" is a "necessary" part of this input/output format compatibility matching concept from the claims. As Implicit has further stated in the reexamination proceedings, this determination is an active one that requires some computing "overhead" to perform; that is, mere passive compatibility is not enough. In other words, the claims require a determination of component format

compatibility, not just that the components somehow happen to be compatible. As the Court further stated:

> The Court also finds persuasive the repeated representations plaintiff made in the reexamination process regarding how the patent uses the format information of the packets to "identify individual components." In particular, "[t]he system of the '163 Patent uses this format information to dynamically identify components necessary for processing the entire message, such that the format of the output data of one module is compatible with the format of the input data of the next module." 9/1/09 Amendment and Response, Hogan Decl, Ex L at 22; see also id. ("Mosberger does not teach to define the input and output formats of the data that is processed by the modules, or to use these formats to make a run-time decision about how to assemble the modules.").
> Markman Order at 10:26-11:5. (Emphasis added)

56. Hence, an accused system will not satisfy this claim limitation unless each processing component providing output data in a particular format provides it in such a format as to be compatible with the required input format for the next processing component in the sequence, and that this input/output compatibility must be determined for each processing component after the first packet is received in order for the component to be included in the assembled processing sequence.

57. I note that for infringement purposes Implicit appears to have adopted a different view, i.e., that this claim limitation may be satisfied if the components simply happen to end up working together, regardless of whether (or when) any affirmative determination of compatibility is made. *See*, *e.g.*, Nettles Report at 36, 49-58 ("The output format of one processing step will match the input format of the next processing step"). It is not clear, however, how Dr. Nettles's position is consistent with the Court's Markman Order.

58. Notably, however, for purposes of invalidity, it appears that Implicit has adopted a view on this point that is closer to what the Court describes in its Markman Order. Specifically, Implicit has stated in the reexamination proceedings that "selecting individual components" requires the system to "ensure that the input format of one component matches the output format of the previous

component" by "adding functionality to check or compare the output format. . .with the input format." *See, e.g.*, 7/10/2012 Implicit's Response in the '857 Reexam at 35; 6/4/2012 Implicit's Response in the '163 Reexam at 20.

4.4.2 Processing

59. The asserted claims of the patents-in-suit use variations on the term "processing" such as "Processing [the] [packets of] [a/each] message," "Processing [the/a plurality of] packets of [a/the/each] message[s]," "Process[ing][es] the next packet of the message," etc.

60. The Court construed the term processing and its variants as used in the claims as "manipulating data with a program." The Court also noted that while the patent claims and specification repeatedly use the term "conversion" when referring to processing, the term should not be limited to "conversion." See Markman Order at 8.

61. For purposes of infringement, Implicit has taken the position that the disclosures of the '163 patent specification concerning packet routing satisfy "processing" without performing a conversion of data.

> Although the conversion system has been described in terms of various embodiments, the invention is not limited to these embodiments. Modification within the spirit of the invention will be apparent to those skilled in the art. <u>For example, a conversion routine may be used for routing a message and may perform no conversion of the message.</u>
> '163 patent at 14:14-19. (Emphasis underlined.)

62. I note, however, that Implicit appears to have taken a different position for purposes of invalidity, namely by asserting in the reexamination proceedings that some of the cited prior art could not invalidate the asserted claims on the basis that "IP router and its components, by definition" do not perform conversion because they handle packets in a "single format," "IP (Internet Protocol) format." *See, e.g.*, 7/10/2012 Implicit's Response in the '857 Reexam at 35; 6/4/2012 Implicit's Response in the '163 Reexam at 20, 30 (attempting to distinguish Kerr on the basis that it does not "change the format of the packet").

4.4.3 "non-predefined sequence of components" and "dynamically identify[ing] a [message specific] sequence of components"

63. All of the asserted claims of the '163 and '857 patents expressly require that the step of "dynamically identifying the sequence of components" occur "after the first packet of the message is received."

64. The Court similarly construed the term "non-predefined sequence of components" as *"a sequence of software routines that was not identified before the first packet of a message was received."* This claim term appears in the '163 patent in claims 1, 15, and 35. For example, claim 1 recites "dynamically identifying a non-predefined sequence of components." See '163 Ex Parte Reexamination Certificate at 1:31-32.

65. In other words, the Court, while allowing for use of a-priori knowledge of components, definitively identified the timeframe during which the component sequence deemed to be non-predefined. The Court emphasized that the claims were self evident: the component sequence cannot be defined in the time period prior to the receipt of the first packet. The identification must therefore take place after the first packet arrival:

> . . .the claim itself identifies the time frame during which the sequence must be non-predefined – i.e., before "the first packet was received."
> Markman Order at 6:17-19.

66. I note also that Implicit has further elaborated on the meaning of dynamically identifying a sequence of components in the current '163 and '857 *inter partes* reexaminations, as well. For example, in its attempt to distinguish Decasper98, Implicit explained that Decasper98 does not dynamically identify a sequence of components because prior to receiving the first packet of the message the order in which packets will pass through components is "predetermined" or "predefined." *See, e.g.,* 7/4/2012 Implicit's Response in the '857 Reexam at 19. In other words, Implicit has taken the position (and the Court's construction confirms) that a system in which the sequential order of components is decided before the first packet of the message is received would

not satisfy the "dynamically identify[ing]. . . a sequence of components" limitations of the asserted claims.

4.4.4 "State Information"

67. The Court construed the term "state information " to be "*information specific to a software routine for a specific message that is not information related to an overall path.*" That is, state information is information that must be maintained on behalf of individual components in connection with the processing of individual sessions.

68. The claim language is explicit in this regard. For example, claim 1 of the '163 patent recites the use of state information by a component for processing a specific "message."

> for each of a plurality of packets of the message in sequence,
> for each of a plurality of components in the identified non-predefined sequence,
> retrieving state information relating to performing the processing of the component with the previous packet of the message;
> performing the processing of the identified component with the packet and the retrieved state information; and storing state information relating to the processing of the component with packet for use when processing the next packet of the message
> '163 Patent Claim 1 (Ex Parte Reexamination Certificate at 1:46-56.) (Emphasis added.)

69. As construed by the Court, the term "message" has the meaning "a collection of data that is related in some way, such as a stream of video or audio data or an email message." See Markman Order at 13:5-6.

70. The Court also determined that it was not sufficient for state information to be just related to the overall path defined by the component sequence. The Court agreed that the patent applicant had explicitly narrowed the meaning of the claims such that the limitations related to state could only be met through state information "not information related to an overall path."

> What the parties do not agree on is defendants' attempt to add, as a further limitation, "not related to an overall path." To support their additional limitation, defendants rely on Implicit's 9/1/09 Amendment and Response where Implicit – <u>in distinguishing Mosberger – argued that "claim 1 is directed to a method in which state information for a specific component is stored on a component-by-component basis</u> and is not information related to an overall path, as the Office Action describes Mosberger." 9/1/09 Amendment and Response, Hogan Decl., Ex, L at 24.

> Plaintiff, in Reply, asserts that the Amendment and Response was simply pointing out that the "novel element" the claim is directed to is the component-by-component rather than overall path storing, and that Implicit was not adding a limitation that the state information could not also relate to the overall sequence or path. Reply at 14. The Court, however, finds Implicit's statement in the reexamination was clear. State information is "not information related to an overall path." This is consistent with the way state information is actually used in the claims and consistent with other language in the claims. See, e.g., '163 Patent Col. 1:48-50 ("retrieving state information relating to performing the processing of the component with the previous packet of the message"); 1:54-56 ("storing state information relating to the processing of the component with packet for use when processing the next packed of the message"); but see Col. 1:39-42 ("storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message"). State information, therefore, is construed as "information specific to a software routine for a specific message that is not information related to an overall path."
>
> See Markman Order at 13:17-14:9. (Emphasis added.)

5 THE ACCUSED PRODUCTS

5.1 Differences in Juniper Products

71. As indicated above (see paragraph 6), Implicit has accused fourteen distinct Juniper products of infringement in this case. However, Dr. Nettles's report fails to provide any individual analysis of these products on a product-by-product basis. The "elements" section of Dr. Nettles's report (see Appendix A) does not even provide any distinction between different types of SRX products (branch vs. high end) or between SRX and J series products. In fact, Dr. Nettles appears to rely on documents and source code pertaining to products that are not even accused of infringement in this case.

72. It appears that Dr. Nettles improperly assumes that all of the Juniper accused and non-accused products function in the same manner so long as they run some form of the Juniper "JUNOS" operating system. The only support Implicit provides for this conclusion are the conclusory statements of Dr. Nettles and a few citations to Juniper's marketing documents promoting the power of a single operating system.

73. Dr. Nettles's reliance on these marketing documents is misplaced and his conclusion is incorrect. These marketing materials (and Juniper's engineers) explain that JUNOS is a single

operating system only in the sense that there is a "single source code base." JUNOS OS: The Power of One Operating System, at 4. (http://www.juniper.net/us/en/local/pdf/brochures/1500059-en.pdf.); *see also* 6/19/2012 Tavakoli Tr. at 150:30-151:3, Dyckerhoff Tr. at 13-28.

74. A single operating system "does not mean is that all of that software is loaded on each and every system." Tavakoli Tr. at 151:4-19; *see also* 9-20-2012 Narayanaswamy Tr. at 109:5-8. To the contrary, as the JUNOS code base is large and complex, with a variety of pieces that get used for Juniper's different products, so one naturally finds many differences between the products themselves. Dyckerhoff Tr. at 19:3-9 ("But Junos is a very big and complex operating system with many differences between the different products we have, and so, you know, what exactly is shared or not, that's the engineer's job. But if you look at our product portfolio, there's many different things in there, and those work differently inside of the code base."). Dr. Nettles does not even acknowledge that any such differences exist, let alone provide a reasoned explanation for why those differences should not be material to his infringement analysis. Indeed, I would expect just the opposite. For example, the claims are very specific in the type of state information that must be used, as well as the nature of the storing and processing each component must perform for the different types of state information. Thus, even slight variations in product functionality may easily be the difference between infringement and non-infringement in this case.

5.2 CPCD Plugin (cpcd_data.c)

75. Dr. Nettles's reliance on the CPCD plugin is one example of the disconnect between his alleged supporting evidence and the Implicit allegations of infringement as to specific Juniper products. Although Dr. Nettles's relies heavily on the source code for the CPCD plugin in an attempt to support his infringement opinions for the accused products, this plugin is not actually even used in the SRX and J series products.

76. Juniper plugins are used in certain products to provide additional, specialized functionality. For example, the CPCD plugin ("Captive Portal Content Delivery") offers the ability to perform IP packet forwarding by re-writing the destination IP address.

77. Juniper uses an approach common in computer programming known as "MAKE" to compile the source code building blocks for its products into executable object code. Each Juniper product package contains the process executable and compiled shared objects that make up a complete process. To do this, Juniper defines a series of files known as "Makefiles." These files may interlate to each other by providing a list of directories containing a dependent Makefile. The MAKE build system processes these dependencies by parsing and then processing the nested Makefiles. Each Makefile processed may also contain a variable holding directory path specification for source files to be compiled and linked in the build.

78. If Dr. Nettles had wanted to demonstrate that the source code he analyzed for the CPCD plugin was actually used in any of the accused products, I would have expected him to provide some evidentiary support or analysis such as an identification of a Makefile or series of Makefiles providing for the Redacted file to be included in a build for an SRX or J series product. Dr. Nettles does not do this. In fact, other than the single Redacted file, Dr. Nettles cites to no Juniper technical documentation at all regarding CPCD functionality.

79. I understand that it is Implicit that bears the burden of proving infringement by a preponderance of the evidence. Fundamental to this obligation is the necessity of providing some basis to believe that the source code and other evidence allegedly supporting Dr. Nettles's infringement analysis actually pertain to the accused Juniper products. Having failed to satisfy this basic threshold requirement, Dr. Nettles's opinions regarding infringement necessary fail.

80. Moreover, although Juniper has no obligation to do so, I have performed an analysis of the relevant Makefiles (which Dr. Nettles failed to do). This analysis further supports my opinion that the CPCD plugin is not used in the accused products.

81. Thus, for example, I looked at the Makefile for the SRX daemon Redacted [1] Notably, the Makefile refers to a number of directories relating to various plugins, but makes no reference to the directories for the CPCD plugin relied upon by Dr. Nettles. Redacted

82. Redacted

83. I went even further and reviewed what are known as the release "manifest" files for the accused products. For example, the manifest file Redacted appears to identify a number of plugins and their configuration file locations. I note that while this manifest file refers to a number of plugins listed in the original Makefile for the flowd daemon build, the CPCD plugin used in Dr. Nettles's analysis is not listed. I also examined numerous other manifest files such as the junos-Redacted and could not find any indication that the CPCD plugin is deployed. Indeed, I did not find in any of the manifest files and Makefiles that I reviewed any indication that the CPCD plugin is used for any of

[1] The "flowd" daemon is the "flow-based forwarding process" for forwarding packets through the device. JUNOS Software Security Configuration Guide at 786.

the SRX or J-Series products. These findings along with a number of other factors related to the build process discussed above lead me to conclude that the use of the CPCD plugin is limited to Juniper's router products (such as the M-Series Router Multiservices PICs) as discussed further in this section of my report.

5.3 "Multiservices" handling

84. The analysis above could also be applied to the other source code files that Dr. Nettles cites and analyzes in his report Redacted Again, Dr. Nettles provides no indication that any of these source code files are actually used in any of the accused products. Accordingly, Dr. Nettles fails to satisfy one of the basic requirements for his analysis to actually provide any meaningful support for his opinions regarding infringement.

85. Again, although not obligated to do so, I have researched the matter and have seen a number of affirmative indications that call into question whether the packet handling code that Dr. Nettles cites is actually used in the accused products. One need not go very far in Dr. Nettles's analysis to see this. For example, the very name of the Redacted Based on my research I have concluded that neither the SRX series products nor the J series products are capable of using the Multiservices PIC. As another example, the path for a number of the code files that Dr. Nettles relies upon shows that those files reside in the Redacted . Again, as explained above, the Multiservices PIC add-on card for Juniper routers (such as the M series) are no longer accused products in this case. Moreover, the Makefile to build the Redacted

Redacted

I also saw code comments suggestive of application to a PIC component as opposed to the SRX or J series products, which do not utilize PIC hardware.

86. Other evidence confirms these conclusions. For example, Dr. Nettles admits that the source code he relies upon permits both flow-based and packet-based processing. See Nettles App. A at ¶¶ 31-32. But he also goes on to contend that certain of the accused products (e.g., the High-End SRX series) do not have the capability of performing packet-based processing. Assuming that this is true, then the source code that Dr. Nettles cites cannot be part of these products.

5.4 Service sets

87. Dr. Nettles's reliance on the concept of a "Service Set" further undermines the validity of his source code analysis. The "Service Set" concept is the starting point for Dr. Nettles's flow-based processing analysis and an essential part of his allegation that the accused products perform the step of "dynamically identifying a non-predefined sequence of components." *See, e.g.*, Nettles Report (Appendix A at pg. 37). And yet the Redacted designed for use with the Multiservices PIC. Reviewing the Juniper documentation confirms that a "Service Set" exists only in the non-accused products and not in the Juniper products accused by Implicit.

88. Juniper documentation regarding Service Sets demonstrates that this is a concept designed for its router products (e.g., M series router and Multiservices PIC). See, e.g. http://juniperpodcast.com/techpubs/en_US/junos10.3/information-products/topic-collections/solutions-guide-session-border-control/index.html?topic-39521.html ("A service set lets you combine rules for different services into one set and then apply the set of services to a service

interface pool on a MultiServices PIC or MS-DPC. You need to configure a service set for each PIC or DPC.") Tellingly, service sets are described in the Juniper treatise on router products (see, e.g., JUNOS Enterprise Routing at 653-657 and 662-666), but not in the Juniper treatise on security products (see Junos Security). And a search for the term "service set," and/or "plug-in" in the Juniper manuals that Dr. Nettles cites in his report did not turn up any hits relating to the SRX and J series products (although I found a number that related to the M and T series routers).

89. In reviewing Dr. Nettles's flawed reliance on the concept of Service Sets, a review of the use of "policies" in the Juniper products is instructive. A policy is essentially a predefined set of instructions that permits, denies, or tunnels specified types of traffic unidirectionally between two points. Policies allow Juniper products to deny, permit, reject, encrypt and decrypt, authenticate, prioritize, schedule, filter, and monitor the traffic attempting to cross from one security zone to another. The CLI Editor is used to configure a set of predefined rules that grouped together as a policy for a particular service. The rules are applied to the processing of different IP packets passing through an interface.

90. A policy applies the rules to the transit traffic within a context (from-zone to to-zone). Each policy is uniquely identified by its name.

> Each policy consists of:
>
> - A unique name for the policy.
> - A set of match criteria defining the conditions that must be satisfied to apply the policy rule. The match criteria are based on a source IP address, destination IP address, and applications.
> - A set of actions to be performed in case of a match—permit, deny, or reject.
> - Accounting and auditing elements—counting, logging, or structured system logging.

OS Security Configuration Guide Release 10.2 at 117.

91. As noted earlier, each policy contains a set of predefined rules that apply to a particular service. A sample policy statement takes a form such as:

```
user@host# set security policies from-zone green to-zone red policy
abctopublic
match source-address abc
user@host# set security policies from-zone red to-zone green policy
abctopublic
match destination-address public
user@host# set security policies from-zone red to-zone green policy
abctopublic
match application ssh
user@host# set security policies from-zone red to-zone green policy
abctopublic
then permit
```

See OS Security Configuration Guide Release 10.2 at 117. (Emphasis added.)

92. Policy lists are searched from top to bottom, so more specific policies are positioned nearer to the top of the list. The action of the first policy that the traffic matches is applied to the packet. If there is no matching policy, the packet is dropped.

93. For certain Juniper products such as routers using Adaptive Services or Multiservices PICs ("AS PIC" or "MIS PIC")[2], Multiservices PICs, and Adaptive Services Modules, the configuration hierarchy involves a different or additional architecture. See JUNOS Enterprise Routing at 396-397, 653-657, 662-666. The system is configured using what are called "service sets." This form of configuration applies to the Juniper M-series routers (for example). As discussed above, the configuration of service sets involves the concept of policies containing rules to define the processing that occurs for a particular service on an interface. A policy may be defined and associated with a rule.

[2] AS PIC (or "ASP") refers to "Adaptive Services Physical Interface Card." See JUNOS Enterprise Routing at 396.

```
[edit services ipsec-vpn]
aviva@RouterA# set ike proposal ike-proposal authentication-method rsa-signatures
```

Create an IKE policy:

```
[edit services ipsec-vpn ike policy digital-cert-policy]
aviva@RouterA# set proposals ike-proposal
aviva@RouterA# set local-id fqdn RouterA.mycompany.com
aviva@RouterA# set remote-id fqdn RouterB.mycompany.com
aviva@RouterA# set local-certificate local-entrust
```

Then, create a rule for a bidirectional dynamic IKE SA that references the IKE policy:

```
[edit services ipsec-vpn rule digital-cert-rule]
aviva@RouterA# set term ike then remote-gateway 10.0.15.2
aviva@RouterA# set term ike then dynamic ike-policy digital-cert-policy
aviva@RouterA# set match-direction input
```

Define a service set for IPSec:

```
[edit services service-set digital-cert-service]
aviva@RouterA# set ipsec-vpn-rules digital-cert-rule
aviva@RouterA# set ipsec-vpn-options local-gateway 10.1.15.1
aviva@RouterA# set ipsec-vpn-options trusted-ca entrust
aviva@RouterA# set next-hop-service inside-service-interface sp-1/2/0.1
aviva@RouterA# set next-hop-service outside-service-interface sp-1/2/0.2
```

See JUNOS Cookbook at 122.

94. The rule is then added to the service set. For example, as illustrated below, the service ***"ipsec-vpn"*** is edited to add a user-defined policy called ***"digital-cert-policy"*** having specific attributes. Then the rule named ***"digital-cert-rule***" is added to "ipsec-vpn" service definition and associated with the policy ***"digital-cert-policy."***

95. Finally, the IPSec service set ***"digital-cert-service"*** is created and defined to contain the rule ***"digital-cert-rule."*** This takes place in the statement "set ipsec-vpn-rules ***digital-cert-rule***" to set the attribute "ipsec-vpn-rules" of service set ***"digital-cert-service"*** to the previously defined rule***"digital-cert-rule."***

96. These predefined configuration steps are then entered into the current operational configuration using the ***commit*** command.

97. In summary, the service set functionality does not infringe the patents-in-suit for a host of reasons, which will be discussed in detail below. But independently of those reasons, Dr. Nettles

has not shown (and cannot) that this functionality is even part of the accused products in this case. Accordingly, his infringement analysis fails before it even reaches the starting line. My understanding is that Dr. Nettles cannot simply assume that all of the Juniper products are the same and thereby shift to Juniper the burden to show that is not the case.

98. Finally, I have not cited all of the examples where Dr. Nettles's report relies on evidence that does not actually correspond to any currently accused product. These examples are not limited to source code. For example, Dr. Nettles repeatedly cites to a guide corresponding to what is known as the Juniper "stand-alone" IDP product. See, e.g., Nettles App. A at ¶ 74; see also IDP Series, Concepts and Examples Guide at 3. But the stand-alone IDP product is not an accused product in this case, and Dr. Nettles has cited no evidence suggesting that the manner in which the stand-alone device monitors and analyzes data is the same or comparable to anything that happens in the SRX or J series products.

6 ALLEGED ACTS OF INFRINGEMENT

99. Before presenting my detailed, element-by-element analysis of the asserted claims, I wish to respond to a number of points raised in what Dr. Nettles calls the "acts section" of his report (pp. 23-34). In my opinion, Dr. Nettles has failed to show any act of direct or indirect infringement by Juniper or third parties. This failure of proof is an independent basis to find non-infringement in this case, separate and apart from the technical element-by-element deficiencies that are presented later in this report.

6.1 Juniper Does Not Directly Infringe

100. I understand that, in order for Implicit to show direct infringement by Juniper, it is not enough to rely merely on hypothetical use or configuration of the accused products in accordance with the claims. Yet that is effectively all that Dr. Nettles has done in his report.

101. For example, Dr. Nettles claims that Juniper infringes through testing of its products. But the asserted claims in this case are not generic claims on any kind of testing; they require specific steps and instructions in order for infringement to occur.

102. To begin, Dr. Nettles admits that the majority of the accused Juniper products (Branch SRX and J series products) can be operated in a non-infringing manner (what he calls "packet-based mode"). I agree that any sort of testing in this mode cannot infringe the patents-in-suit.[3]

103. However, Dr. Nettles is wrong to go on to suggest that testing in what he calls "flow-based mode" will necessarily infringe the patents-in-suit. Again, the asserted claims in this case comprise numerous detailed elements; the fact that a product may operate in a "flow-based mode" is not sufficient to show infringement. Implicit's corporate designee, Mr. Balassanian, conceded this point in his deposition: " Q. So if I tell you that Company X has a stateful, flow-based firewall, have I told you everything you need to know to have a good-faith belief that Company X infringes the '163 patent? A. No." Balassanian 08-16-2012 Depo Tr. at 985:5-9.

104. Dr. Nettles seems to merely assume that testing in accordance with the elements of the asserted claims must occur within Juniper. But if this testing were so ubiquitous as Dr. Nettles claims, it should have been easy for him to point to some source code or other documentary evidence that Juniper actually did this. He simply fails to do so. Indeed, his (unsupported) allegation that Juniper tests "essentially all aspects of the accused products" for "each release" concedes that there are at least some aspects that are not tested.[4] He also fails to show that any

[3] Moreover, if the software code that Nettles cites were in fact representative of all of the accused products, then this argument would apply to high-end SRX products also.

[4] Starting in 2010, Junos releases are numbered according to a year.sequence convention. Thus, using the example of 10.4R2.6, the first part 10.4 is the major version number (the 4th release of the year in 2010). R is the type of release, in this example the R designates a shipped release (other types are S for service release and B for a Beta release). 2 is the revision number, i.e. the numbered sequence of the maintenance releases, where the First Release Shipment (FRS) is always R1. And 6 is the build number (also known as the spin number) of the release. The spin number indicates the specific build of the release and may be requested by JTAC during service discussions. In his

testing takes place in the United States (the job search website that Dr. Nettles cites does not satisfy this requirement).

105. Dr. Nettles also speculates that it is "likely" that Juniper uses the accused products in its own networks. Proof of infringement cannot be founded on this type of speculation, and in any event, Dr. Nettles still presents no evidence that any such internal use is performed with the necessary element-by-element steps or instructions.

106. Dr. Nettles goes on to opine that Juniper directly infringes by performing tests, configuration, or other assistance with its customers. But again, there is no element-by-element analysis or proof in support of this claim. The fact that a Juniper product was configured as a "firewall" or in a "stateful" mode or "flow-based mode" is not evidence of infringement, as Implicit itself admits.

107. For alleged customer-related acts, Dr. Nettles does not opine about the allocation of work performed by Juniper versus its customers. Without this information, Dr. Nettles has failed to show that the allegedly infringing acts satisfy the legal requirements regarding "divided infringement," such as control by Juniper, etc.

108. Nor has Dr. Nettles shown any direct use of the features and configurations that he specifically relies upon to show the detailed elements of the claims. For example, Dr. Nettles's infringement positions rely heavily on theories drawn from the behavior of the CPCD plugin. Yet I note that Dr. Nettles has not provided any evidence of any Juniper employee or customer ever using the CPCD plugin. Therefore, he has not satisfied the burden of proof required to prove infringement based on the CPCD plugin. The same can be said of the other technical details upon

---

report, Dr. Nettles examined only code relating to release 11,1R2.3 and has provided no allegation or analysis as to how this release is different or the same as other releases. Thus, he has failed to satisfy Implicit's infringement burden with respect to products based on other releases.

which Dr. Nettles purports to rely for his infringement opinions: use of session interest functions or session ignore messages, application identification caching, service sets, reconfiguration at runtime, addition of plugins to the Juniper product, etc. Without actual evidence of these (direct or even circumstantial), Dr. Nettles cannot prove direct infringement by Juniper.

109. As another example, in paragraph 31 of his report, Dr. Nettles purports to provide an illustration of "basic packet processing" in the Juniper products. But he fails to establish when and under what conditions this process is invoked. Assuming for the sake of argument the applicability of the source code that Dr. Nettles cites, it appears that before proceeding with any of the outlined steps, Redacted

Yet Dr. Nettles has provided no evidence of any such actual configuration. See also Introduction to Service PICs at 9-10 ("A policer, filter, service filter, service set, postservice filter, and input forwarding-table filter are applied sequentially to the traffic; these are all optional items in the configuration."). Without this information, Dr. Nettles fails to show direct infringement.

6.2 Juniper Does Not Indirectly Infringe The Asserted Claims

110. Dr. Nettles has likewise failed to present evidence of direct infringement by others that might form the predicate for a claim of indirect infringement by Juniper. For Juniper to be found to induce infringement, the user(s) operating the Juniper product must perform all of the steps of a claim, or satisfy all of the elements of a system claim. Furthermore, Implicit must demonstrate that Juniper had specific intent to cause infringement.

111. For Juniper to be found to be a contributory infringer, Implicit must show that Juniper made, used, or sold a material component of a patented invention which is not itself a staple article of commerce and with knowledge that the component is especially made for use in an infringement of a patented invention. As discussed below, Juniper does not satisfy these requirements, either.

6.2.1 Inducement

112. Dr. Nettles states in his report that:

> I have been asked to assume the Juniper had specific intent to induce infringement of the patents-in-suit and/or was willfully blind. Nevertheless, it is my opinion that Juniper knew or should have known that its actions with respect to the Non-Juniper Direct Infringers would lead to actual infringement. At a minimum, Juniper should have known from a review of the '163 and '857 patents that both itself and the Non-Juniper Direct Infringers were infringing the' 163 and '857 patents.
>
> Nettles Report at ¶90.

113. Of course, Dr. Nettles cannot show induced infringement based on a mere assumption of intent. And in any event, there is no evidence of such specific intent on Juniper's part. Although Dr. Nettles claims that "Juniper has put forth no credible non-infringement defenses," in fact this report sets out numerous such defenses. I understand that Juniper also reasonably believes that the patents-in-suit are invalid—a belief that appears to be shared by the Patent and Trademark Office based on the current status of the reexamination proceedings (i.e., all asserted claims stand rejected).

114. Dr. Nettles also cites to a handful of passages (mostly deposition transcripts) that he says shows intent. But at most this evidence shows mere knowledge on the part of Juniper regarding certain aspects of customer configuration, as opposed to active inducement. Certainly generic "training courses" and "instruction manuals" do not satisfy this requirement.

115. There is also no indication that unnamed information about alleged customer configurations bears upon the specific individual limitations of the patents-in-suit. Nor has Implicit produced any actual configuration evidence from Juniper's document production or source code tree, or that of any Juniper customer. And the reliance here on "Matte and Brill depositions" is inexplicable, as I am aware of no such depositions in this case.

116. Again, there is no evidence that Juniper assists customers with CPCD plugins or any of the other specific features identified as matching particular claim elements in Dr. Nettles's report. Nevertheless, I presume that if Juniper were to assist in the installation or configuration of a CPCD plugin, it would strive to do so in a manner that would not generate the types of error messages that Dr. Nettles contends may infringe the asserted claims. In other words, even under Implicit's theory of infringement, it is most logical to conclude that Juniper would encourage its customers not to "infringe."

6.2.2 Contributory Infringement

117. Dr. Nettles further opines that Juniper is liable for contributory infringement on the theory that Juniper systems are supposedly not suitable for substantial non infringing use:

> The Juniper accused products are a component (or entirety) of the patented machine, manufacture, combination or composition, including the claimed computer media, and a material or apparatus for practicing the patented methods here, and are especially made or especially adapted for use in an infringement of the patents-in-suit, and not staple articles or commodities of commerce suitable for substantial non infringing use.
>
> See Nettles Report at ¶98

118. This conclusion is contradicted by Dr. Nettles's own report, in which he admits that the majority of the accused products contain an admitted non-infringing use: packet-based mode. And in fact, there are a host of additional non-infringing uses for the accused products even under Implicit's theories of infringement. For example, any use that omits use of the CPCD plugin, Redacted , application identification caching, etc. would not infringe under Implicit's theories. These uses are substantial; they are real-world, practical modes of operation. Use of plugins that merely inspect (but do not manipulate data) or that do not retrieve, use, and store state information as set forth in the claims, are likewise substantial non-infringing uses even under Implicit's view.

119. The Juniper products are general-purpose, configurable devices (of course, configurability does not equate to infringement), and in that sense they are staple articles of commerce not especially adapted for infringement.

7 '163 PATENT CLAIM ANALYSIS

120. In my opinion, Dr. Nettles has failed to show by a preponderance of the evidence that the Juniper Accused Products infringe any of the asserted claims of the patents-in-suit. This section presents an analysis addressing on a claim-by-claim and element-by-element basis a number of the deficiencies underlying Dr. Nettles's conclusions regarding infringement.

121. For convenience in responding to Dr. Nettles's report, I will address the portions of the claims as he has broken them down in his report. Nevertheless, I note that, in a number of cases where Dr. Nettles has grouped multiple elements together, he ultimately fails to show that each and every individual element and limitation is met (even though he concedes, as he must, that infringement of a patent claim requires satisfaction of each and every element of the claim). See Nettles at ¶25.

122. Note that there are a number of non-infringement issues that are common to more than one asserted claim (or even to more than one element within a given claim). In the interest of brevity, I have not necessarily repeated my analysis for each of these below, but rather intend to incorporate by reference any analysis applicable to identical or similar claim terms or issues wherever they appear (regardless of whether or not I have expressly included language indicating incorporation by reference).

7.1 '163 Patent Claim 1

7.1.1 "1. A method in a computer system for processing a message having a sequence of packets" (Preamble)

123. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

124. For example, Dr. Nettles has not identified what he considers to be the "message" used in the Juniper Accused Products that would satisfy the claim language. This is not a trivial omission; an alleged "message" as claimed provides the necessary context for many subsequent claim elements, such as limitations regarding steps to be taken "for the first packet of the message" or "for each of a plurality of packets of the message." These limitations are discussed in further detail below.

125. As another example, Dr. Nettles has also not provided any evidence of "processing" a message, as discussed further below.

7.1.2 "the method comprising: providing a plurality of components, each component being a software routine for converting data with an input format into data with an output format" (Element 1a)

126. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.1.2.1 Dr. Nettles Has Not Provided Evidence of "a plurality of components"

127. One issue with respect to the term "components" must be addressed as an initial matter. The Court has construed the term "components" to simply mean "software routines." Markman Order at 5. Nevertheless, it is not clear how Implicit and Dr. Nettles intend to apply that construction in practice, or whether their positions regarding this claim term are even consistent with each other. For example, I have reviewed Implicit's 7/25/12 response to Juniper's Interrogatory No. 18 (including claim charts), in which Implicit states that the "Kerr" reference

"**does not discuss any sort of software routines**," although Implicit admits the reference discloses "processing" and "treatments" of packets in "multiple processing stages." Yet Dr. Nettles opines that the Juniper Accused Products include "components" because they contain a "series of actions (modules)." Nettles ¶ 72. It is not clear how these positions are reconciled or what additional requirements Implicit considers might be part of the meaning of "software routines." Should Implicit or Dr. Nettles explain further and/or persist in advancing a narrow application of this term at trial, I reserve the right to provide additional opinions and analysis on this point with respect to my non-infringement analysis.

128. In any event, Dr. Nettles has not shown "a plurality of components" as claimed here. I understand that Implicit was required by the Court to provide code citations on a limitation-by-limitation basis to support Implicit's infringement accusations. See 5/23/12 Order. In response to this Order, Implicit provided code citations only for the CPCD plugin in its supplemental infringement contentions. Similarly, the report of Dr. Nettles is supported by source code cites for just the CPCD plugin alone; no code cites for any other plugins or other alleged "components" are provided. No other evidence pertaining to a plurality of plugins has been provided that is supported by detailed analysis of source code.

129. The claim language at issue here requires a "plurality of components," that is, more than one. This same requirement is reflected in other aspects of the claims, e.g., the plural "components" or the term "sequence of components." As such, I understand that Implicit must provide evidence showing how a plurality of components satisfy the claim limitations. In my opinion Implicit has not done this. Dr. Nettles has limited his source code supported analysis to a single plugin, namely the "CPCD" plugin. He fails to provide evidence of how the claim

limitations are satisfied because he does not analyze the interaction of a plurality of plugins, nor does he provide evidence to support his positions using the source code provided by Juniper.

130. Dr. Nettles does refer in passing to a "list of plug-ins," which is attached as an exhibit to his report:

> Also see Exhibit 3 for a list of plug-ins. Thus in my opinion JNI's accused products meet limitation 1a.
>
> Id. (Emphasis added.)

131. In my opinion, the so-called "list of plug-ins" provided in Dr. Nettles's Exhibit 3 does not support his conclusion regarding "plurality of components." Dr. Nettles provides support for only one of the plugins on this list (CPCD, or "junos-cpcd"), and even then only refers by reference to the corresponding source code citations ("see also 1a-1g below"). However, for the other purported plugins listed in Exhibit 3, there is literally no analysis or description whatsoever. There is no citation to source code for any of the other plugins, much less any analysis of their roles, properties, technical characteristics, manner of operation, etc. To establish that claim requirement that these plugins be embodied in "software routines" (as opposed to, for example, ASICs or other hardware), Dr. Nettles's should have been expected to at a minimum identify the corresponding software routines in the source code. But he failed to do even this.

132. None of the other material cited in this section of Dr. Nettles's report identifies any alleged "components" other than CPCD plugin. Dr. Nettles cites to three lines of source code (analyzed further in the section below), none of which appear to require the use of multiple plugins in the Juniper Accused Products.

133. Furthermore, in the paragraphs ¶¶72-75 offered as support for this claim limitation, Dr. Nettles merely alleges that the Accused Products "perform IPS algorithm processing" and quotes sections from two publications regarding IPS/IDP. But there is no citation to or analysis of any

Juniper code relating to IPS/IDP technology. And indeed, Implicit's corporate designee plainly testified that "IPS is not a component" for purposes of this claim element. Balassanian Depo. 1169:2. In short, this section of Dr. Nettles's report is simply a reproduction of Juniper's published materials without explanation. As a result, Dr. Nettles's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

134. Therefore Dr. Nettles cannot demonstrate that Juniper's systems provide a "plurality of components" according to this step of claim 1 of the '163 patent.

7.1.2.2 Dr. Nettles Has Not Provided Evidence of Components That Satisfy "converting data with an input format into data with an output format"

135. Even if Dr. Nettles's analysis had sufficiently defined a "plurality of components," he has failed to demonstrate that "each component" is a "software routine for coverting data with an input format into an output format." Dr. Nettles has not provided any evidence to support this limitation but merely provides an unsupported conclusion:

> As discussed below, in the JNI technology overview section, and throughout this report, each module, operating at a specific network layer and performing certain processing, has an input format and an output format.
>
> Nettles Appendix A at ¶71. (Emphasis added.)

136. Elsewhere in this section, Dr. Nettles likewise claims (without support) to have identified components "for converting data." Id. ¶ 70.

137. In the rest of this section, however, Dr. Nettles fails to provide any supporting analysis for these claims. He does not identify or describe the data formats of plugins he identifies, or what conversion is allegedly performed. If Dr. Nettles believed (for example) that the plugins listed in Exhibit 3 satisfied this claim limitation, one would have expected him at a minimum to identify for each plugin ("each component"): (1) the input format, (2) the output format, and (3) the conversion process for going from the input format to output format. But in fact, Dr. Nettles never identifies

this information, nor does he provide any substantive source code analysis or technical analysis from which this information could be derived.

138. Similarly, the "evidence" section of his report is simply a reproduction of Juniper's published materials without explanation. As a result, Dr. Nettles's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

139. The cited "evidence" provided, including sections marked as "Evidence '163 C1 1a(1)," "Evidence '163 C1 1a(2)," and "Evidence '163 C1 1a(3)" do not discuss how the plugins "convert[] data with an input format into data with an output format." Id. at 73-74. None of this evidence describes conversion processes or the input/output formats of plugins.

140. Dr. Nettles provides an initial cite to source code at 1044:399. (Nettles Appendix A at ¶29.) However, as seen in the cited excerpt below, this reference provides no insight as to any conversion process from an input format to an output format.

Redacted

141. The code he refers to simply iterates through a data structure containing plugin identifiers (and indeed, there need not be more than one plugin). There is no reference to input or output formats, and the cited code certainly performs no conversion from one format to another.

142. Then Dr. Nettles cites to source code at 1068:1319 and 1070:1383 (Nettles Appendix A at ¶29.)

Redacted

See JSC00001068:1318-1320.

Redacted

See JSC00001070:1383-1385.

143. Once again, Dr. Nettles does not venture close to the subject at hand, namely the claim limitation "providing a plurality of components, each component being a software routine for converting data with an input format into data with an output format." Rather, he appears to address a completely different subject of how plugin processing may begin based on function calls corresponding to two different event types. His analysis sheds no light on the determination of input and output formats, and (again) there is no indication that this code performs any conversion of data from one format to another.

144. Therefore, Dr. Nettles cannot demonstrate that the Juniper Accused Products provide components, "each component being a software routine for converting data with an input format into data with an output format."

7.1.3 "for the first packet of the message, dynamically identifying a non-predefined sequence of components for processing the packets of the message such that the output format of the components of the non-predefined sequence match the input format of the next component in the non-predefined sequence" (Element 1b)

145. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.1.3.1 "Dynamically Identifying a Non-Predefined Sequence Of Components"

146. Dr. Nettles has not provided evidence of "dynamically identifying a non-predefined sequence of components."

147. The Court concluded that the term "non-predefined sequence of components" should be construed as *"[a] sequence of conversion routines that was not identified in or determinable from configuration information in place before the first packet of a message was received.*"

148. As for the term "dynamically identifying," the Court found that no construction was necessary because the term was "expressly defined in the claims themselves." Order at 7. "For

example, Claim 1 provides that 'wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received.'" Id. This interpretation is, of course, consistent with the "non-predefined" construction; the critical point in time is the moment the first packet of a message is received, and the identification step must occur after this moment.

149. Put another way, the Court stated that, based on statements Implicit made during the first reexamination proceeding, Implicit had disclaimed the use of pre-configured paths. See Markman Order at 6:8-10.

150. Redacted. See Nettles Appendix A at ¶¶44-50. Moreover, the handful of Juniper source code files and other materials that Dr. Nettles analyzes show a process for handling packets that follows a predefined pattern (established by what he claims is the "service set") that is configured and loaded before the first packet in a session. See discussion in section 6.2.3. For convenience, I will refer to this functionality as described in these Juniper materials as the Juniper "Service Set" functionality.

151. The Service Set functionality provides a predefined sequence defining a path through which network packets may flow. The appropriate preconfigured path in any given instance is chosen upon receiving the first packet of a packet sequence. In this sense, the Service Set functionality does exactly the opposite of what is claimed in the patents.

152. Dr. Nettles nevertheless attempts to present a number of arguments in an attempt to satisfy the "dynamically" limitation. He first argues that the "script-like directives" contained in pre-configured Juniper policies "direct the system to identify the processing components"

> Based on that inspection, the accused products utilize a technique of "policy expressions," which are script-like directives that are loaded and re-loaded into the systems while they are running. They may be loaded and re-loaded into the systems by people, other systems or software, or both.

> The policies direct the system to identify the processing components and algorithms that are to be applied to the network traffic that is classified through the packet inspection.
>
> Nettles Appendix A at ¶84. (Emphasis added.)

153. Dr. Nettles appears to incorrectly opine that the Juniper policies specify non-predefined processing sequences merely because there are many and varied policies to consider.

> Fully custom traffic/application flow specifications, as well as fully custom processing components, can be dynamically loaded and re-loaded into the system as well. Because of the configurability of policy expressions, and traffic/applications specifications, there are near infinite resultant processing sequences – non-predefined – which will execute.
>
> Nettles Appendix A at ¶85. (Emphasis added.)

154. The claim itself and the Court's claim construction ruling make it clear, however, that the claim element is only satisfied if the components in the sequence are selected after the detection of the first packet. This cannot be true for component processing sequences that are, as Dr. Nettles agrees, derived from policies "loaded and re-loaded into the systems by people, other systems or software, or both" prior to detection of the first packet. See Nettles at ¶ 85. In fact Dr. Nettles is essentially in agreement that the Juniper policies are pre-defined, not "dynamically" chosen in a "non-predefined" manner as the claim requires.

155. Dr. Nettles has provided no substantive technical analysis of how the Juniper plugins could satisfy this claim limitation. The cited "evidence" provided, including sections marked as Evidence '163 C1 1b(1) through Evidence '163 C1 1b(18). Id. at 85-89. None of Dr. Nettles's cites describe a mechanism for "dynamically identifying a non-predefined sequence of components for processing the packets of the message." This section of his report is simply a reproduction of Juniper's published materials without explanation. As a result Dr. Nettles's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

156. Dr. Nettles refers to source code in the following cites for support for his opinions on "Creating a Data Processing Path Based on Information in the First Packet."

Redacted

157. The cites above provide no substantive analysis on how the Juniper plugins are ordered into a sequence based on the processing of an initial packet. For example, the Redacted at JSC00001138, line: 481 simply calls a function Redacted Dr. Nettles has not indicated the functionality in Redacted nor its dependence on the number of plugins involved. Similarly, the cite at JSC00001121, line: 827, for Redacted

158. Tellingly, Dr. Nettles's references to the critical items (6) and (7) in his list above identify no supporting evidence and provide no pinpoint code citations. Indeed, even setting aside the lack of supporting evidence, the description itself borders on the nonsensical; it is difficult to even understand what is meant by the statement Redacted And later, Dr. Nettles opines about something called a "plugin mask," but even then is inconsistent about when it is purportedly created. *See* Nettles App. A at ¶ 35, ¶ 39.

159. In short, none of Dr. Nettles's listed "steps" provide any insight into the claim limitation of "dynamically identifying a non-predefined sequence of components."

7.1.3.2 Service Sets Do Not Satisfy the Court's Construction for "dynamically" and "non-predefined"

160. As explained above, the Court's Markman Order makes it clear that a "non predefined" sequence of components must be "dynamically" determined after the receipt of the first packet. By contrast, the Juniper Service Set functionality uses a concept of predefined service sets to define the handling that occurs for a particular service on an interface.

161. Within the Service Set functionality, an administrator defines and groups service rules by configuring the service-set definition. A defined service set is then associated with an interface. In turn, a policy may be defined and associated with a rule. The rule can then be added to the service set. For example, as illustrated below, the rule named "digital-cert-rule" can be added to the IPSec service set "digital-cert-service."

```
[edit services ipsec-vpn]
aviva@RouterA# set ike proposal ike-proposal authentication-method rsa-signatures
```

Create an IKE policy:

```
[edit services ipsec-vpn ike policy digital-cert-policy]
aviva@RouterA# set proposals ike-proposal
aviva@RouterA# set local-id fqdn RouterA.mycompany.com
aviva@RouterA# set remote-id fqdn RouterB.mycompany.com
aviva@RouterA# set local-certificate local-entrust
```

Then, create a rule for a bidirectional dynamic IKE SA that references the IKE policy:

```
[edit services ipsec-vpn rule digital-cert-rule]
aviva@RouterA# set term ike then remote-gateway 10.0.15.2
aviva@RouterA# set term ike then dynamic ike-policy digital-cert-policy
aviva@RouterA# set match-direction input
```

Define a service set for IPSec:

```
[edit services service-set digital-cert-service]
aviva@RouterA# set ipsec-vpn-rules digital-cert-rule
aviva@RouterA# set ipsec-vpn-options local-gateway 10.1.15.1
aviva@RouterA# set ipsec-vpn-options trusted-ca entrust
aviva@RouterA# set next-hop-service inside-service-interface sp-1/2/0.1
aviva@RouterA# set next-hop-service outside-service-interface sp-1/2/0.2
```

See JUNOS Cookbook at 122.

162. See discussion in section 6.2. Dr. Nettles claims that "an ordered list of the Plugins to be executed is obtained from the Service Set."

> Dynamically identifying a nonpredefined sequence of components:
> **1049:596** (An ordered list of the Plugins to be executed is obtained from the Service Set)
> **1068:1319** (Run the chain of plugins assigned to this Service Set with "Session Interest" message and the packet being processed)
> **1051:650** (For every plugin from the list obtained, call Plugin's 'data handling' function [**1053:726**], and analyze the result)
> **1054:763** (exclude plugin from future processing of the packets of the current session if 'session ignore' return code has been returned by the plugin)
>
> Example:
> **1004:185** (An example of CPCD plugin handling first packet)

Nettles Appendix A at ¶37.

163. But this "ordered list" is specified as part of the Service Set prior to the first packet processing, because someone such as an administrator must have set it up that way in advance (as shown in the example above). As Dr. Nettles himself states, the "ordered list" he identifies already exists before the first packet and is merely obtained during the processing of his identified "first packet." Therefore, use of that same ordered list cannot constitute a "dynamic" identification of a "non-predefined" sequence of components.

7.1.3.3 The "session_ignore" Error Codes Do Not Satisfy the Court's Construction for "a non-predefined sequence of components"

164. Perhaps recognizing the fundamental defect in his infringement theory regarding the Juniper Service Set functionality, Dr. Nettles attempts to rely on one other aspect of the cited Juniper code in support of his opinion. This additional argument is based on a summary of the process for handling packets by the "Captive Portal Content Delivery" ("CPCD") plugin. For example:

Redacted

Nettles Appendix A at ¶51.

165. Here Dr. Nettles appears to identify the Redacted function as a mechanism for cataloging error returns from the CPCD plugin, presumably on the (mistaken) theory that skipping the CPCD plugin could somehow be equivalent to establishing a brand new sequence of components after the first packet. There are a number of problems with this theory. As an initial matter, as explained above, the CPCD plugin is the only alleged "component" implicated in Dr. Nettles's infringement theory; if that plugin is skipped altogether, then it is unclear how there could possibly be a sequence of components as required by the claims.

166. I further note that the function Redacted reports only runtime or configuration error conditions. The conditions that result in an Redacted Redacted are determined by misconfiguration or operational error conditions, not by logic that determines whether the CPCD plugin should contribute to processing a message. Redacted

167. For example, some of the CPCD plugin functions that potentially result in the return value Redacted

168. If required resources are not available because of misconfigured policies or a lack of system resources, the program determines that the CPCD can not perform services it was designed for. The function calls that test for availability of resources include (Redacted

169. The remaining conditions that, if ever encountered, would give rise to the Redacted return value correspond to functionality that is not implemented in this version of the CPCD plugin based on the cpcd rule action specified in the cpcd policy. As such, those error returns would never be encountered unless the administrator had incorrectly configured the system to perform functions it was not designed for.

170. Notably, Dr. Nettles fails to prove that the error conditions giving rise to the Redacted error return have ever in the past or will ever in the future occur. Indeed, if the Service Set functionality is properly configured and used, they should not occur.

171. Moreover, the Redacted error return is not a design feature to support the dynamic identification of a non-predefined sequence of components. The Juniper design intends for the CPCD plugin to contribute to the correct functional behavior of the Juniper device, specifically the re-direction of packets to the requesting party. If the CPCD plugin reports a Redacted error then the system is not working as it was originally intended, that is, packets do not get re-directed to the intended IP address.

7.1.3.4 The Redacted Error Codes Cannot Satisfy the Limitation of "dynamically identifying a non-predefined sequence of components"

172. Even if a CPCD plugin or other plugin identified in a predefined service set were to be skipped due to an error condition, this would not change the sequence of the other plugins (if any) in that service set.[5] Indeed, Dr. Nettles has not identified any source code that could cause this sequential order to change after the first packet in response to a session ignore message or any other reason. Nor have I seen any source code in my review that could perform this function. As Implicit stated in the reexamination proceedings, dynamically creating the processing path involves both determining which components are necessary and in what order they should be applied. In the case of the Service Set functionality, the order is fixed in advanced and not changed by anything that happens after the first packet. Moreover, the Dr. Nettles theory of removing plugins from a predefined list is in fact the logical opposite of dynamically selecting individual components, in the sense that the latter adds and the former subtracts.

7.1.3.5 Dr. Nettles Fails To Identify the "first packet of the message"

173. In his report on infringement, Dr. Nettles has failed to identify a "first packet" and thus failed to show that the accused Juniper instrumentalities perform all of the requisite steps "for the first packet of the message." I have also looked at other materials and statements from Implicit provided during the discovery phase of this case on this point, including Implicit's responses and supplemental responses to Juniper Interrogatory No. 20, as well as certain briefing to the Court regarding this interrogatory (and the Court's order). Specifically, Implicit stated that: "the first packet is the packet that creates a message specific, stateful data processing path as per the claims. Depending on device configuration, this can be the first control packet, used in establishing the

[5] I have reviewed the technical materials cited in Dr. Nettles's report, including manuals and other papers taken from the Juniper web site, and I have not seen anything suggesting the ability to change the order of plugins to be applied to a message after the first packet of that message is received.

TCP/IP handshake, or the first packet containing content (the more typical case) - it depends on configuration." See Implicit Brief, Dkt. No. 140.[6] The Court went on to state that Implicit would be held to this broad "functional" definition for purposes of its infringement theory in this case. See Order, Dkt. No. 144.

174. Both of the two "first packet" examples that Implicit cites create serious problems in terms of Implicit's infringement theory. For example, Implicit suggests that a TCP connection request containing the TCP synchronize flag ("SYN") set for a connection request can be considered the "first packet" of a message.[7] However, these TCP handshake messages are not processed by the alleged Juniper sequence of components, as required by the claims. On the other hand, if Implicit believes the limitation is satisfied by the "first packet containing content," then it is not clear that, at that point, the flow or session has not already been established by the system. In other words, in that case, the alleged sequence of components would be created (even under Implicit's view) before the first packet is received, not after.

175. Implicit states that identification of the "first packet" in a particular allegedly infringing scenario "depends on configuration." It is unclear what configuration Implicit is referring to here. My understanding is that Dr. Nettles has had a full opportunity to review any configuration information he wanted, but he does not cite any in his report to support the "first packet" limitation. In any event, regardless of configuration information, I do not see how the claims can be satisfied under any scenario falling within the "functional" identification that Implicit has provided.

[6] It is not clear what is meant by "content" here (Layer 4 content? Layer 7 content?) and Dr. Nettles does not explain further.

[7] See e.g., the TCP handshake as disclosed in RFC 793. See also Computer Networks (Tanenbaum 2003) at 529-533.

7.1.3.6 The Accused Juniper Products Provide IP-To-IP Packet Processing and Cannot Satisfy "Selecting the Individual Software Routines of the Sequence so That the Input and Output Formats of the Software Routines Are Compatible."

176. Claim 1 of the '163 patent requires that the step of identifying a sequence of components be performed "such that the output format of the components of the sequence match the input format of the next component in the sequence." This same concept is captured in all of the other asserted claims of the patents in suit. For convenience, I refer to this requirement as the "input/output matching" limitation.

177. The Court has expressly stated that determining the compatibility between the output of one software routine and the input of the next is a necessary part of "selecting individual components." See Markman Order at 11.[8] Minimally, this requires that the Juniper accused instrumentalities perform some comparison of the output capabilities of one plugin with the input capabilities of another. For example, one way to do this would be to compare data structures output and input by plugins. Determination of compatible plugin formats would then involve comparing the input and output format of the data structures to see if they were identical or otherwise compatible with each other. I have reviewed the source code Dr. Nettles cites and found no such "determination" as required by the Court's Markman Order.

178. The '163 patent teaches that identifying a sequence of compatible conversion routines requires examining components for format compatibility between layers of a protocol stack such as Ethernet/IP/TCP:

> "[w]hen a packet of a message is received, the conversion system ... identifies a sequence of conversion routines [i.e., creates a "path"] for processing the packets of the message by comparing the input and output formats of the conversion routines." '163 patent at 2: 40-45. This is done by examining the packets of the message layer by layer (where the layers might be Ethernet/IP/TCP as described in the previous paragraph), using a mapping process to identify the

[8] Implicit's corporate representative also confirmed agreement with the Court' statement. See Balassanian Depo. at 1001:1-8, 1002:1-2 ("Q. Do you agree with that statement? A. I -- I do. Sure. Yes.").

> sequence that should be associated with the message, and then creating the message-specific sequence of processing components (i.e., the path). '163 patent at 3:65-5:4.
>
> Implicit Response to Request For '163 Patent Re-exam at 9.

179. Implicit further explained this claim limitation in statements to the PTO in the ongoing re-examination for the patents-in-suit. Specifically, Implicit took the position that "single format" operation such as IP packet processing and IP router functionality would fall outside the scope of the claims of the '163 patent.

> Kerr discloses a system and method for packet handling within an IP router. Kerr never discloses or discusses the format of the data packets processed by its system. Presumably, this is because Kerr limits its disclosure to an IP router and its components, which, by definition, are only designed to handle packets in the IP (Internet Protocol) format—a single format. Nor is there any evidence that the "treatments" in Kerr change the format of the packet. Accordingly, Kerr assumes that all packets will be in IP format, and therefore that the input and output of every component will be a packet with an IP format. Given the single format of data (IP) that Kerr processes, it teaches away from considering format compatibility when "dynamically identifying a non-predefined sequence of components."
>
> Id. at 29-30. (Emphasis added.)
>
> The Examiner's inherency argument fails in large part because Decasper simply has no need for the claimed limitation. Decasper's architecture is specifically designed to handle only one format: Internet Protocol (IP). In fact, adding functionality to check or compare the output format of a packet processed by one IP component with the input format required by the next IP component would only add costly overhead, since all of Decasper's components use IP and thus no input/output matching between components is necessary.
>
> Implicit Response to Request For '857 Patent Re-exam at 21. (Emphasis added.)

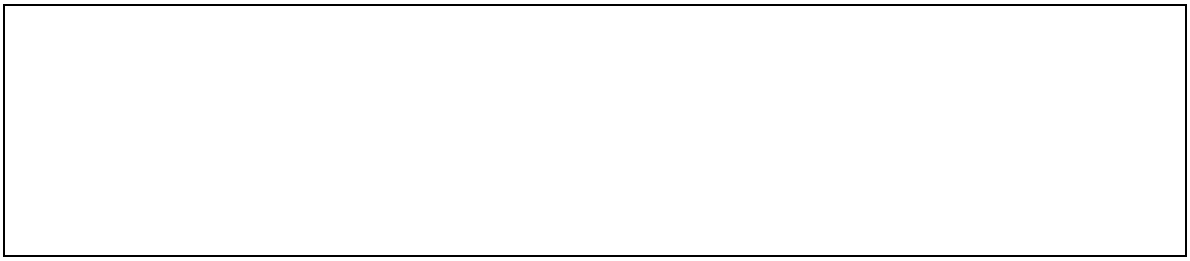
180. In these statements, Implicit once again emphatically categorizes IP packets as packets having a single format and therefore not capable of satisfying the '163 patent claims. According to Implicit, processing that simply involves manipulation of IP packets cannot satisfy the claims of the patents-in-suit because "the input and output of every component will be a packet with an IP format." In other words, "no input/output matching between components is necessary" when all the components involve handling or manipulation of IP-formatted packets. Implicit also concedes that this determination is not mere passive compatibility; some affirmative checking is required because

Implicit contemplates that adding input/output matching capabilities could add "overhead" to a system (e.g. processing bandwidth involved in performing the compatibility check).

181. Dr. Nettles has not addressed infringement beyond the single CPCD plugin module. However, as can be readily established from the CPCD source code, the only processing performed is on either the header or payload of an IP Packet. The CPCD plugin does not add headers to these packets (i.e., an Ethernet header) nor does it remove the IP header. Thus, any input to the CPCD plugin, or output from the CPCD plugin "will be a packet with an IP format."

182. Redacted

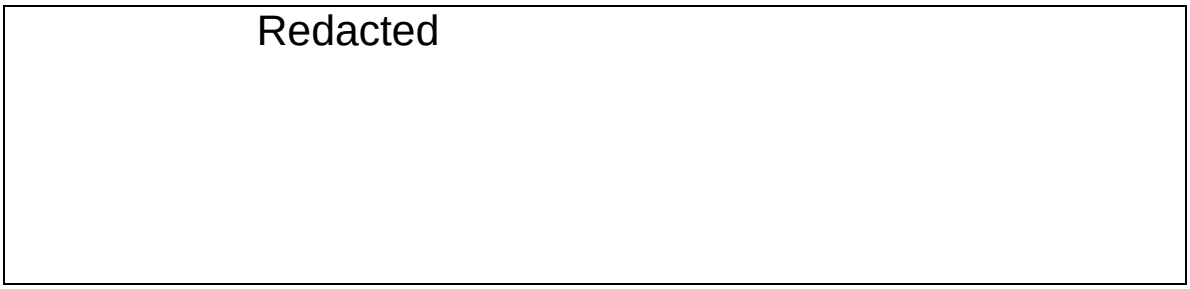
Redacted



See SC00001015 at 598-606.

183. Inside this function, which processes packets following the Redacted

Redacted



Redacted

See SC00001013/14 at 497-539.

184. As can be seen from the code in Redacte the processing contained in function Redacted See SC00001009:354 - SC00001011:430.

185. Redacted

186. According to Implicit, this represents a system that teaches away from the claims of the patents-in-suit.

> An IP router and its components, by definition, are only designed to handle packets in the IP (Internet Protocol) format—a single format. Accordingly, the Decasper architecture assumes that all packets will be in IP format, and therefore that the input and output of every component will be a packet with an IP format. Given the single format of data (IP) that Decasper processes, it teaches away from considering format compatibility when "dynamically identifying a non-predefined sequence of components."
>
> Id. at 20. (Emphasis added.)

187. The accused Juniper instrumentalities do not select software routines by determining the compatibility between the output of one software routine and the input of the next. To the contrary, the accused Juniper instrumentalities are used in a manner that merely assume IP packets, such that there is no evidence that the plugins depend on any input or output constraints other than IP.

188. Furthermore, the input/output matching limitation further supports the notion that the claim requires multiple components in a sequence as opposed to just one. After all, if a sequence of components could be a single component, then no input/output matching between components would be necessary. As explained above, Implicit here identifies only a single alleged component, i.e., the Juniper source code with filename Redacted (see JSC00001000), which is insufficient to establish a plurality of components in a sequence. Having failed to identify multiple allegedly infringing components, it also follows that Dr. Nettles cannot have identified any supposed mechanism for performing input/output matching between multiple components.

189. Dr. Nettles has not provided any evidence of that the Juniper plugins he identifies provide for such a determination of input/output format compatibility.

> As discussed below, in the JNI technology overview section, and throughout this report, each module, operating at a specific network layer and performing certain processing, has an input format and an output format. In the JNI technology overview section see in particular Flow-based Processing from Junos Enterprise Routing, Flow-based Processing based on Junos source code, and JNI's basic packet processing loop. Also see Exhibit 3 for a list of plug-ins. Thus in my opinion JNI's accused products meet limitation 1a.
>
> Nettles Appendix A at ¶71. (Emphasis added.)

190. In the sections cited Dr. Nettles fails to analyze and/or describe the data formats of plugins he identifies. Dr. Nettles has provided no substantive technical analysis of how the Juniper plugins could provide input/output format specifications that would be used for compatible selection of plugins. For example, he simply makes conclusory statements such as the passage above and later "[t]he output format of one processing step will match the input format of the next processing step." See Nettles Appendix A at ¶85.

191. Similarly, the "evidence" section of his report is simply a reproduction of Juniper's published materials without explanation. As a result Dr. Nettle's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

192. The cited "evidence" provided, including sections marked as "Evidence '163 C1 1a(1)," "Evidence '163 C1 1a(2)," and "Evidence '163 C1 1a(3)" do not discuss how the Juniper plugins could provide input/output format specifications that would be used for compatible selection of a plugin. Id. at 73-74. None of this evidence describes the input/output formats of plugins nor does it describe the compatibility of various plugins. It also does not provide evidence of how Juniper plugins would satisfy the claimed "components" in such a way as to allow compatibility to be determined from the components.

193. As explained above, I have also carefully reviewed the source code excerpts that Implicit cites in support of this claim limitation. In order to satisfy this claim limitation, Dr. Nettles ought to have identified some portion of code that examines input and output formats between plugins and determines their compatibility. Not only does Dr. Nettles fail to do this, but I have examined the source code files themselves and have found no evidence of input/output matching. For example, there is no indication that a Juniper product would skip a plugin and then check to see if the preceding and subsequent plugins are format-compatible.

194. Therefore, Dr. Nettles cannot demonstrate that Juniper's systems satisfy the limitation of identifying a component sequence such that "the output format of the components of the non-predefined sequence match the input format of the next component in the non-predefined sequence" according to this step of claim 1 of the '163 patent.

7.1.4 "wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components after the first packet is received" (Element 1c)

195. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

196. For example, the "dynamically identifying" limitation has been discussed in significant detail above. Any alleged identification and ordering of plugins—even under Dr. Nettles's

theory—would be performed before the first packet is received, not "after" (as required by the claims).

197. As another example, as the Court has stated (and Implicit has conceded) a necessary part of "selecting individual components" includes the input/output matching functionality. Dr. Nettles likewise fails to show that this aspect of the claims is satisfied, as explained above.

198. As another example, Dr. Nettles again has failed to prove or identify a sequence of components (plural), but rather limits his analysis to a single plugin, namely the CPCD plugin. See discussion above.

7.1.5 "and storing an indication of each of the identified components so that the non-predefined sequence does not need to be re-identified for subsequent packets of the message" (Element 1d)

199. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

200. For example, based on the citations that Dr. Nettles cites as support for this limitation, it appears he believes that the accused Juniper instrumentalities merely store (if anything) Redacted, which is different from a sequence of plugins to use for subsequent packets of the message. This is explained in detail above.

201. As another example, Dr. Nettles claims that certain functions associated with the CPCD plugin can result in the Juniper systems skipping the CPCD plugin for the entirety of a particular session. But even under Implicit's theory, the same or similar error functions could just as easily result in a decision to skip the CPCD plugin part-way through a session. Such operation is contrary to the claim language that requires that the operative sequence need not be "re-identified" for packets following the first packet of an alleged message.

202. It is not even clear where Dr. Nettles believes the check is performed to determine whether a packet is a first packet or subsequent packet. See Nettles App. A at ¶ 31-39.

203. As another example, as explained above, Dr. Nettles has failed to identify any alleged components other than CPCD, and this limitation clearly requires multiple components.

7.1.6 "and for each of a plurality of packets of the message in sequence, for each of a plurality of components in the identified non-predefined sequence, retrieving state information relating to performing the processing of the component with the previous packet of the message" (Element 1e)

204. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

205. As an initial matter, this step must be performed "for each of a plurality of packets of the message in sequence," and "for each of a plurality of components in the identified non-predefined sequence." Dr. Nettles fails to satisfy this requirement. He does cite in this section a few lines of source code that he contends disclose a "[p]lurality of packets" and a "[p]lurality of components." See Evidence '163 C1 1e(1). But this claim limitation does not merely require the existence of a plurality of packets and components (which the cited source code fails to disclose in any event). Rather, Dr. Nettles was required to show that the "retrieving" step be performed for each of a plurality of packets and components. This he fails to do. And without such analysis, Dr. Nettles cannot satisfy Implicit's burden of demonstrating that this limitation is satisfied.

206. Moreover, the Juniper systems do not store and retrieve state information for implemented plugins that pertains to the processing of the previous packet in a sequence of packets that makes up a specific message. As noted in section 5.4.4, the Court construed the term "state information " to be "*information specific to a software routine for a specific message that is not information related to an overall path.*" That is, state information is information that must be maintained on behalf of individual components in connection with the processing of individual

sessions. The construction requires that state information pertaining to the component processing of a packet that is part of a sequence of packets in a session must be held for that component so that the subsequent packet in the same session can use the state information for that component pertaining to the previous packet.

207. Dr. Nettles makes the purely conclusory statement that this limitation is met by the Juniper products:

> It is my opinion that in JNI's accused products retrieve state information relating to performing the processing of the component with the previous packet of the message for each of a plurality of packets of the message in sequence and for each of a plurality of components in the identified non-predefined sequence. As is evident from the JNI documents, deposition testimony, code, and other evidence cited herein, JNI's accused products meet the limitation. . . .
> See below, the JNI technology overview section, the 1f and 1g discussion. In the JNI technology overview section, see in particular Flow-based Processing from Junos Enterprise Routing, Flow-based Processing based on Junos source code, JNI's basic packet processing loop, and Running the Code Modules (Plugins)
>
> Nettles Appendix A at ¶116.

208. Dr. Nettles has provided no substantive technical analysis of how the Juniperplugins could satisfy this claim limitation. The cited "evidence" provided, includes sections marked as Evidence '163 C1 1e(1) through Evidence '163 C1 1e(3). Id. at ¶117. None of Dr. Nettles's cites describe a mechanism for "retrieving state information relating to performing the processing of the component with the previous packet of the message." This section of his report is simply a reproduction of Juniper's published materials without explanation. As a result Dr. Nettle's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

209. Dr. Nettles appears to incorrectly opine that the Juniper Redacted

Redacted

See Id. at ¶117.

210. Redacted

211. (SC00001357:163.) However, this single line statement provides no evidence whatsoever of the storage and retrieval of state information on a per-plugin basis that is generated on a packet-by-packet basis as a result of message processing.

212. Redacted

213. (JSC00001013:519.) The function cited provides only a pointer Redacted . Dr. Nettles does not bother to investigate the contents of this data structure that is located by the pointer. To establish that the data structure referred to by Redacted is unique to the CPCD plugin (and not "related to an overall path"), Dr. Nettles would have to show that other plugins used different state data structures and that the data structure referred to by Redacted is not used by any other plugin in a defined "component" sequence. However, Dr. Nettles has not chosen to provide such evidence.

214. Moreover, instead of showing "retrieving" for a "plurality of components," as required by the claim, Dr. Nettles merely points to a single plugin, Redacted (see Juniper source code at JSC00001013:519).

215. Dr. Nettles's argument with respect to this claim limitation is also inconsistent with his arguments regarding other limitations in the claim. For example, earlier in his report, Dr. Nettles contends that a sequence of components is "dynamically identified" when a s[Redacted] plugin allegedly causes that plugin to be skipped for a given session. If that is so, then it is not clear how the CPCD plugin can be retrieving, using, or storing state information for that session. And Dr. Nettles never identifies how any source code pertaining to any other alleged component or plugin satisfies this or the other "state information" limitations. It does not seem possible that the CPCD plugin could somehow be skipped but still satisfy these limitations, as Dr. Nettles suggests.

216. Additionally, Dr. Nettles relies on a Juniper feature in which an "application identification" is cached for later use. See IDP Series Concepts and Examples Guide at 96. In his Evidence 163 C1 1e(2) he provides the passage:

> When the application identification feature identifies a new application, it caches the result (the destination address, port, protocol, and service) to reduce processing for subsequent sessions. The application cache and extended application cache are maintained separately.
>
> See Id. at ¶117.

217. However, Implicit's source code citations do not include any reference to the code comprising Juniper's application identification caching functionality, as I understand Implicit was obligated to do per a Court order. Dr. Nettles also fails to prove that Juniper's application identification caching functionality is specific to a particular plugin and message rather than merely related to an overall path.

218. In summary, Dr. Nettles has failed to show that the accused Juniper instrumentalities satisfy this claim element.

7.1.7 "performing the processing of the identified component with the packet and the retrieved state information" (Element 1f)

219. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

220. As explained above, this step must be performed "for each of a plurality of packets of the message in sequence," and "for each of a plurality of components in the identified non-predefined sequence." Dr. Nettles fails to satisfy this requirement. For example, he does not identify a "plurality of components," much less perform the necessary analysis to show that each such component performs "processing" as that term has been defined by the Court ("manipulating data with a program"). Dr. Nettles does not identify any "program" corresponding to each of a plurality of components. Nor does he identify how he believes each such component allegedly manipulates (changes) data in a packet, as opposed to merely inspecting data and passing the packet along. Without such analysis, Dr. Nettles fails to satisfy Implicit's burden of demonstrating that this limitation is met.

221. Redacted and a Juniper feature in which an "application identification" is cached for later use. See IDP Series Concepts and Examples Guide at 96. As explained above, Implicit's source code citations do not include any reference to the code comprising Juniper's application identification caching functionality, and thus Dr. Nettles cannot satisfy Implicit's burden for this claim limitation by reference to this feature.

222. As discussed above in section 8.1.6, Dr. Nettles has identified the contents of the Redacted Redacted the state used in "performing the processing of the identified component with the packet and the retrieved state information." However, his identified state

information does not satisfy the Court's construction for state information, as explained in detail in my discussion of the other "state information" limitations (see, e.g., below).

7.1.8 and storing state information relating to the processing of the component with packet for use when processing the next packet of the message. (1g)

223. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

224. As explained above, this step must be performed "for each of a plurality of packets of the message in sequence," and "for each of a plurality of components in the identified non-predefined sequence." As explained above, Dr. Nettles fails to satisfy this requirement.

225. Redacted and the contents of the data structure referenced by the returned pointer Redacted to satisfy this claim limitation. However, Dr. Nettles fails to identify any "state information" that is stored for use when processing the next packet of the message as required to satisfy the limitation "storing state information relating to the processing of the component with the packet for use when processing the next packet of the message." Redacte simply retrieves the session extension structure and uses values without storing information derived from the packet processing.

226. The processing of packets following the session interest phase is determined by the call to the function Redacted with event type

Redacted

Redacted

227. (JSC00001015:600-602.) The function Redacted

Redacted

JSC00001014:554, and JSC00001014:561. Redacted

(JSC00001013:532.)

228. Redacted JSC00001009:353-1011:430)

Redacted

229. Redacted

"storing state information relating to the processing of the component with the packet for use when processing the next packet of the message." Redacted is not modified for each pass through the packet processing code that follows the processing of the first packet. Dr. Nettles even appears to concede this point in his report, as Redacted as opposed to during each of a plurality of subsequent packets (which is what the claims require). Of course, the first packet cannot satisfy the claim 1 state information limitations, which require retrieving information from a previous pocket. And, Dr. Nettles never explains how any information stored is used for the "next packet."

230. In addition to the source code citations, Dr. Nettles also reproduces block quotes from a Juniper publication regarding (1) an application identification caching functionality as well as (2) a "Profiler" feature. Again, Dr. Nettles fails to provide any explanation of these features, does not cite or analyze any source code relating to these features, and nowhere explains how they might satisfy this claim limitation. As a result, Dr. Nettles's opinions regarding this claim element are conclusory and presented without any scientific reasoning.

231. In summary, Dr. Nettles fails to satisfy Implicit's burden of showing that this or any other limitations of claim 1 of the '163 patent is satisfied by the Juniper Accused Products.

7.2 '163 Patent Claim 15

7.2.1 A method in a computer system for demultiplexing packets of messages, (Preamble)

232. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

233. I note that Juniper's accused products do not perform "demultiplexing" as that term would be understood by one of ordinary skill in the art reading the '163 patent.

234. The '163 patent specification identifies the role of the claimed "demultiplexer" that is responsible for determining the next component to be used in the processing. ("This routine is passed the packet (message) that is received, an address structure, and a path entry structure. The demux routine extends a path, creating one if necessary. The routine loops identifying the next binding (edge and protocol) that is to process the message and "nailing" the binding to a session for the message, if not already nailed." See '163 patent at 8:43-49. Emphasis added..) See also '163 patent 3:36-58 and Figure 8. The result of the claimed demultiplexing operation is that packets are sent to different processing queues. ("[T]he forwarding component may use a demultiplexing ("demux") component . . . .that is to process the packet and then queues the packet for processing by the path. . . . Each path thread is responsible for retrieving packets from the queue of its path and forwarding the packets to the forwarding component." See '163 patent 3:17-24.)

235. Identifying the next processing component therefore involves examination of a packet to determine the processing for the next protocol stack level. The result is a stream of packets being "demultiplexed" into processing queues for the next "edge," i.e., entry into the next protocol stack level, ("The binding is the edge of a protocol." See '163 patent at 8:61.)

236. Implicit's statements in the current '163 and '857 *inter partes* reexaminations further confirm that demultiplexing requires processing for the next protocol stack level. For example, in its response to the first office action in the '163 *inter partes* patent reexamination, the patentee described the "invention of the '163 patent" as an:

> "innovative leap of dynamically creating a path of specially sequenced components after the arrival of the first packet was critical in order to handle both the explosion of protocols (with varying data formats) for network applications **and processing "up" the stack into an ever changing landscape of formats and data types**.

6/4/2012 Implicit's Response in '163 *Inter Partes* Reexamination at 6; 8 ("This processing 'up' the network stack allows a device to extend the network stack into the application layer, providing efficiencies and flexibility that the old static model could not."); 9 (explaining that identifying the proper sequence of components "is done by examining the packets of the message layer by layer (where the layers might be Ethernet/IP/TCP as described in the previous paragraph).").

237. Implicit has further confirmed this meaning during the litigation. For example, in distinguishing the prior art reference Decasper98, the patentee argued that the use of the term demultiplexing in Decasper98 was not the demultiplexing discussed in the patent because Decasper98 is a router, which operates at only one layer in the network stack, the IP layer:

> As explained below, the Decasper router only processes one data format (IP). There is no notion of changing data formats in this reference; it is not a demultiplexing system at all.
> 7/27/2012 Implicit's Sixth Supplemental Responses to Juniper's Second Interrogatories at 35.

238. As discussed in section 8.1.3.6, the processing contained in Redacted

Redacted See SC00001009:354 - SC00001011:430. None of this packet handling represents a demultiplexing of a packet stream into processing queues at a different protocol level. The packet processing is entirely limited to data elements contained within an IP (Internet Protocol) formatted packet. Therefore the accused functionality analyzed by Dr. Nettles does not perform any type of "demultiplexing" as recited by the preamble of claim 15.

239. Thus for the reasons set forth here and above, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.2.2 the method comprising: dynamically identifying a non-predefined sequence of components for processing each message based on the first packet of the message so that subsequent packets of the message can be processed without re-identifying the components (Element 15a)

240. For all the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1b, 1d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.2.3 wherein different non-predefined sequences of components can be identified for different messages, each component being a software routine (Element 15b)

241. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1a, *see also* Element 35a), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.2.4 and wherein dynamically identifying includes selecting individual components to create the non-predefined sequence of components (Element 15c)

242. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1b, 1c), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products. For example, Dr. Nettles fails to show that the input/output matching limitation is met (as shown above).

7.2.5 and for each packet of each message, performing the processing of the identified non-predefined sequence of components of the message (Element 15d)

243. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1e, 1f), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products. Dr. Nettles again fails to show the claim requirements are met for "each packet" (e.g., packets in the TCP handshake, if considered part of the message). Moreover, Dr. Nettles makes no attempt to show that these claim requirements are met for "each message." There could be, for example, some messages that receive CPCD handling, while others do not. The claim language, by contrast, requires that the claimed method always be performed for "each message."

7.2.6 wherein state information generated by performing the processing of a component for a packet is available to the component when the component processes the next packet of the message (Element 15e)

244. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1g), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.3 '163 Patent Claim 35

7.3.1 A computer-readable medium containing instructions for demultiplexing packets of messages, (Preamble)

245. In my opinion, Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

246. I note that Juniper's accused products do not perform "demultiplexing" as that term would be understood by one of ordinary skill in the art reading the '163 patent. See '163 patent claim 15 analysis in section 8.2.1 regarding "demultiplexing."

247. The accused functionality analyzed by Dr. Nettles does not perform any type of "demultiplexing" as recited by the preamble of claim 35.

248. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1b, *see also* Element 15d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.3.2 by method comprising: dynamically identifying a message-specific non-predefined sequence of components for processing the packets of each message (Element 35a)

249. Moreover, for all of the reasons set forth above in my discussion of the corresponding element(s), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

250. In addition, Dr. Nettles fails to show that the alleged sequence of components is "message-specific." Moreover, as explained in this report, Implicit has failed to provide its contention as to what it believes the message in the Juniper system is. Accordingly, it is difficult to understand what meaning this term take in the context of the accused Juniper products. I, therefore, reserve the right to respond at a later date for at least those claim elements using the term "message-specific" should Implicit address what it believes is a "message" in the Juniper system.

7.3.3 upon receiving the first packet of the message wherein subsequent packets of the message can use the message-specific non-predefined sequence identified when the first packet was received (Element 35b)

251. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1b, 1d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.3.4 and wherein dynamically identifying includes selecting individual components to create the message-specific non-predefined sequence of components (Element 35c)

252. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1b, 1c), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.3.5 and for each packet of each message, invoking the identified non-predefined sequence of components in sequence to perform the processing of each component for the packet (Element 35d)

253. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Elements 1e, 1f), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.3.6 wherein each component saves message-specific state information so that that component can use the saved message-specific state information when the component performs its processing on the next packet of the message (Element 35e)

254. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1g), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products. For example, Dr. Nettles admits that the Juniper products can be configured to "allow [ ] multiple plugins to share the same session." Nettles Appendix A ¶ 40.

7.4 '857 Patent Claim 1

7.4.1 A method in a computer system for processing packets of a message, (Preamble)

255. For all of the reasons set forth above in my discussion of the corresponding element(s), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.4.2 the method comprising: receiving a packet of the message and a data type of the message (Element 1a)

256. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Elements 1a, 1b), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

257. Moreover, Implicit has not identified an alleged data type. Notably, Implicit stated in the '857 *inter partes* reexamination that the "Invention of the '857 Patent" requires format change and "processing 'up' the stack into an ever changing landscape of formats and data types."

258. I note also that Implicit has yet to take a concrete position on the meaning of the term "data type" as applied in the context of the Juniper system. Thus, I reserve the right to respond at a later date at least to those claim elements containing the term "data type" should Implicit address what it believes is the "data type" in the Juniper system.

7.4.3 analyzing the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence (Element 1b)

259. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Element 1b), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.4.4 wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received (Element 1c)

260. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Element 1c), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.4.5 storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message (Element 1d)

261. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Element 1d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.4.6 for each of a plurality of components in the identified sequence: performing the processing of each packet by the identified component (Element 1e)

262. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Elements 1e, 1f), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.4.7 and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message (Element 1f)

263. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., '163 patent, Element 1g), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5 '857 Patent Claim 4

7.5.1 A method in a computer system for processing a message, the message having a plurality of headers, (Preamble)

264. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1a), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5.2 the method comprising: analyzing the plurality of headers of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence (Element 4a)

265. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1b), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5.3 wherein analyzing the plurality of headers of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received(Element 4b)

266. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1c), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5.4 storing an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message (Element 4c)

267. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5.5 for each of a plurality of components in the identified sequence: performing the processing of each packet by the identified component (Element 4d)

268. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1e), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.5.6 and storing state information relating to the processing of the component with the packet for use when processing the next packet of the message (Element 4e)

269. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1f), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6 '857 Patent Claim 10

7.6.1 A computer readable storage medium, other than a data transmission medium, containing instructions for processing packets of a message, the instructions comprising at least one computer-executable module configured to: (Preamble)

270. For all of the reasons set forth above in my discussion of the corresponding element(s), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.2 receive a packet of the message and a data type of the message (Element 10a)

271. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1a), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.3 analyze the data type of a first packet of the message to dynamically identify a sequence of components for processing a plurality of packets of the message such that the output format of the components of the sequence match the input format of the next component in the sequence (Element 10b)

272. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1b), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.4 wherein analyzing the data type of the first packet of the message to dynamically identify the sequence of components includes selecting individual components to form the sequence of components after the first packet of the message is received (Element 10c)

273. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1c), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.5 store an indication of each of the identified components so that the sequence does not need to be re-identified for subsequent packets of the message (Element 10d)

274. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1d), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.6 for each of a plurality of components in the identified sequence: perform the processing of each packet by the identified component (Element 10e)

275. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1e), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

7.6.7 and store state information relating to the processing of the component with the packet for use when processing the next packet of the message (Element 10f)

276. For all of the reasons set forth above in my discussion of the corresponding element(s) (e.g., Element 1f), it is my opinion that Dr. Nettles has failed to demonstrate that this portion of the claim is satisfied by the Juniper Accused Products.

8 ALLEGED VALUE OF IMPLICIT'S PATENTS

277. I also disagree with Dr. Nettles's conclusion regarding the alleged value of the patents-in-suit. Dr. Nettles alleges that the "packet processing technology invented by Implicit has significant value," because it is a "fundamental feature" of the accused products' "flow-based"

processing. Nettles Report at 34. The cornerstone of Dr. Nettles's conclusion is that "stateful, flow-based processing" is vital to the accused products and that Juniper's customers would not buy the Juniper products absent this technology. Nettles Report at 34-40.

278. Dr. Nettles, however, improperly equates Implicit's purported invention with maintaining a flow table session. Indeed, Dr. Nettles asserts that "remembering (or 'caching')" of information related to an overall path is "central to infringement" and to the value of Implicit's patents. Nettles Report at 40-41 ("I know of no alternative to caching the results of the first packet computation that would provide acceptable performance."). Maintaining a session table for flows or "caching" is not, however, Implicit's invention. In fact, the maintenance of a session table or "caching" was well known many years before the priority date of the patents-in-suit. Former Balassanian collaborator David Wolf made this precise point in his deposition testimony:

> Q. And you mentioned a technique of caching where basically, after you do a complicated process, you store the results for later so that you don't have to perform that complicated, expensive process again and again. Do you recall that?
> A. Yes.
> Q. That caching concept, is that something that was well known in computer science in the 1980s and early 1990s?
> A. Yes.

Wolf Tr. at 99.

279. Dr. Nettles also contends that Juniper's stateful, flow-based processing "is central to the infringement" and that there are "no acceptable non-infringing alternatives" and "no alternative to caching the results of the first packet computation that would provide acceptable performance." Nettles Report at 40-41. I disagree. Dr. Nettles's conclusions are incorrect.

280. Indeed, Edward Balassanian, the purported inventor of the patents-in-suit, has admitted that the patents-in-suit are not needed to perform stateful flow-based processing, and that stateful flow-based processing can be implemented in a non-infringing manner. Balassanian 30(b)(6) Tr. at 985:5-989:7.

281. Thus, as Mr. Balassanian himself acknowledges, there are alternative ways to implement the accused functionality. This admission completely undermines the central premise underlying Dr. Nettles's conclusion regarding the "value" of the claimed invention. Mr. Balassanian testified at length that alternative approaches to his patents existed in the prior art.

282. For example, in his deposition on 8-16-2012, he testified products could be built that did not in his opinion infringe the claims of the Patents-In-Suit, including (1) products that support the Ethernet protocol (Id. 1208:7-25), (2) products that support the IP protocol (Id. 1208:1-7), (3) products that implement firewalls with zones, screens, and policies (Id. 1208:8-20), (4) products that that implement intrusion-prevention functionality (Id. 1208:24-1209:14), (5) products that implement antivirus protection (Id. 1210:8-17), (6) products that implement Network Address Translation (Id. 1210:18-1211:3), (7) products that implement Web site access blocking (Id. 1211:13-22).

283. Again in his deposition testimony on 8-16-2012, Mr. Balassanian testified that products could be built that did not in his opinion infringe the claims of the Patents-In-Suit, including products that implement a stateful flow-based firewall consisting of a firewall that proxies TCP connections, i.e., a stateful, flow-based firewall. Id. 985:5-25. He also testified that a "stateful flow-based firewall" could be either infringing or non-infringing:

> 25 So you could have a stateful, flow-based
> 1 firewall that practices the claims of the '163 patent;
> 2 correct?
> 3 A Sure. Yes.
> 4 Q And you could also have a stateful, flow-based
> 5 firewall that does not practice the claims of the '163
> 6 patent; correct?
> 7 A Correct.
> Id. at 988:25-989:7

284. Similarly, in his deposition testimony on 8-16-2012, Mr. Balassanian testified that his invention was not a patent covering routers and firewalls generally:

10 Q So it wasn't as if, for example, the '163
11 patent issued and you thought to yourself, I now have a
12 patent on the underlying technology for today's routers
13 and firewalls?
14 A I'm sure what I thought was that I had a patent
15 that was specifically described by the claims of that
16 patent. That's what I thought. That's what I invented.
17 I didn't invent routers and firewalls, as we have
18 discussed for five days previously.
081612EBalassanian Depo Tr. at 974:10-18.

285. During the deposition carried out under oath on May 30, 2012, Mr. Balassanian provided additional testimony regarding product implementations that were outside the scope of his patents. 12-05-30 Balassanian Depo Tr. For example he stated that he was not the inventor of the network router. Id. 55:9-12. He also conceded that was not recognized as the inventor of the proxy firewall or the intrusion detection system. Id. 55:13-58:9. He further testified that the concepts of looking at application layer packet data was well known at the time of his patent applications:

25 Q Prior to your invention of the
1 patents-in-suit, did the concept exist of looking
2 at application layer data as part of a network
3 security device?
4 MR. HOSIE: Objection. Vague, ambiguous,
5 overbroad. 10:45:21
6 THE WITNESS: Virtually all operating
7 systems that are on a network can be considered a
8 network security device, and they have applications
9 running on them, and they look at application data.
10 By your definition, yes.
Id. at 59:25-60:10.

286. Similarly, Mr. Balassanian provided additional testimony regarding product implementations that existed prior to his patent applications, including "deep packet inspection" that required the examination of application data in a packet. 12-05-30 Balassanian V1 Depo Tr.

10 Q Broadly speaking, deep packet inspection
11 existed before you conceived the patents-in-suit,
12 correct?
13 A If by deep packet inspection you mean
14 looking at all the layers of a packet, every
15 network stack will look at the first layer of a
16 packet, then the second layer of a packet, then the

17 third layer of a packet and ultimately get to the
18 application layer, when an application gets data it
19 can process without having all the other layers on
20 it. That concept is part and parcel to network
21 processing.
Id. at 61:10-21.
. . . . .
1 THE WITNESS: I would -- I honestly don't
2 know. It very well could be that those were post
3 my invention. It doesn't really matter because you
4 can look at application layer data. Every computer
5 with a network stack looks at application layer
6 data.
7 If that's what you mean by deep packet
8 inspection, then that existed before my patent. If
9 what you mean by network security devices that use
10 deep packet inspection, if that means they have an
11 ethernet, IP, TCP, HTTP or other application layer
12 protocol, and that they look at each layer of that
13 packet -- aka inspecting it deeply or deep packet
14 inspection -- and they serve as a network security
15 device, then yes, in that sense, network security
16 devices that use deep packet inspection existed
17 before my patent.
Id. at 65:1-17.

287. Mr. Balassanian testified that firewalls were configurable existed prior to his inventions and that he did not invent "stateful firewalls." 12-05-30 Balassanian V1 Depo Tr.

15 Q And, in fact, there were firewalls that
16 existed prior to the patents-in-suit that could be
17 configured, correct?
18 MR. HOSIE: Objection. Vague, ambiguous,
19 overbroad.
20 THE WITNESS: The concept of a firewall
21 existed before my patent. The concept of
22 configuration existed before my patent.
Id. 46:15-22.

7 Q Are you the inventor of the stateful
8 firewall?
9 MR. HOSIE: Same objections.
10 THE WITNESS: I would not say I'm the
11 inventor of a stateful firewall.
Id.39:7-11.

288. In Implicit's discovery responses, e.g., 2012-06-29 Implicit's Response to First RFA, Implicit admitted that many technology elements or processes disclosed in the Patents-In-Suit were

in existence or practiced before Edward Balassanian conceived the inventions covered by the Patents-In-Suit. For example, Implicit admitted the prior existence of "flow-based processing" in the sense of "identifying flows" (Id. 7:2-13), "deep packet inspection" in the sense of "inspecting contents of a packet for some purpose" (Id. 7:16-22), "stateful packet inspection" in the sense of "general concept of stateful firewalls" (Id. 7:24-8:6-9), "network processing based on packet data (as opposed to headers alone)" in the sense of "a processing system where information other than information in the headers is used" (Id. 8:11-18), "network processing based on data in the application layer" in the sense of "a processing system where information other than information in the headers is used" (Id. 8:20-27), "configurable network security devices" in the sense of "a security device where function could be changed by configuration" (Id. 9:2-15), "the ability to add new policies to a network security device" in the sense of "a firewall that could be configured by a user" (Id. 9:17-25), "the ability to add new plug-ins to a network security device" in the sense of "adding software components to larger software platforms" (Id. 10:1-11), and "the ability to add new policies to a network security device without shutting down the device" in the sense of "a firewall that could be configured by a user" (Id. 10:13-21)

9 NON-INFRINGING ALTERNATIVE TECHNOLOGIES

289. As set forth in detail above, my opinion is that the Juniper accused products do not infringe the patents-in-suit. However, even if the functionality and features that Dr. Nettles identifies in his report were deemed infringing (even though they are not), there are other ways to accomplish the same results in a different way that would not infringe Implicit's patents even under Implicit's view.

9.1 Standard For Non-Infringing Alternatives

290. An acceptable non-infringing alternative is one that : (1) was available in the market place at the time of infringement; or (2) was known to the accused infringer if he/she possessed all the necessary equipment, know-how, and experience to create a non-infringing alternative.

291. Non-infringing alternative system components are perfectly adequate if they satisfy the business requirements of the defendant by, if necessary, performing those requirements in a manner that is different from the patent claims.

9.2 Exemplary Non-Infringing Alternatives

292. In my opinion, given the detailed nature of the elements of the asserted claims, it would be very straightforward to provide suitable non-infringing substitutes for the functionality that Dr. Nettles identifies as infringing.

293. For example, Dr. Nettles points to the use of Redacted functions and Redacted values. As I explain above, these are not intended to be invoked in normal usage but exist to help properly handle error conditions should they occur.

294. This represents one possible design choice for error handling but not the only one. For example, one could simply allow the plugin to execute the error tests every time and not just on the basis of the Redacted first pass through the plugin. In my opinion, any associated performance penalty associated with this approach would be de minimis. Alternatively, one could create multiple service sets and select among them.

295. Dr. Nettles also relies heavily on the CPCD plugin, but there are numerous other ways to achieve the security or other features provided by the CPCD plugin without using the "captive portal" approach. For example, a Juniper Networks customer who used CPCD to authenticate users and protect access to the Internet through a private wireless network (e.g., at a hotel), could as an

alternative allow guests to access to individual wireless access points with passwords distributed to guests via some other business arrangement, such as a request for connectivity at the time of guest check-in. As such, the functionality contained in the CPCD plugin would no longer be required.

296. Alternatives likely exist for any number of the plugins contained in the Juniper accused products. As noted above, Implicit has failed to identify the specific "plurality of components" it alleges infringe the patents-in-suit. Although Exhibit 3 to Dr. Nettles's report identifies a list of plugins, Implicit has failed to identify a plurality of these plugins that it believes can be operated together in an allegedly infringing manner. Nor can Dr. Nettles be contending that any combination of the plugins in Exhibit 3 infringes, since some of the plugins identified cannot be grouped together in the accused products. Without knowing the "plurality of components" Implicit is accusing of infringement, it is not possible to identify the full range of non-infringing alternatives. I reserve the right, therefore, to respond with additional non-infringing alternatives should Dr. Nettles provide this information at a later date.

297. Finally, I note that Implicit has claimed that each and every one of the prior art products and references that Juniper has cited in fact do not practice the claims of the patents-in-suit. To the extent that one or more of these prior art systems teaches a system that is outside the scope of the claims, then these prior art approaches could serve as adequate non-infringing alternatives. See e.g., '163 Patent Request For Inter Partes Reexamination dated February 13, 2012 and '857 Patent Request For Inter Partes Reexamination dated March 2, 2012. This is not surprising, as Implicit's accusations of infringement are directed at old technology that predates the patents-in-suit. Indeed, the accused functionality (stateful flow-based processing), was available well-before the priority date of the patents-in-suit.

298. Assuming, as Implicit claims, that none of the prior art references Juniper has cited practice the claims of the patents-in-suit, it would have been relatively simple for Juniper to operate in a manner that Implicit claims is non-infringing by using some or all of these prior art techniques. Although Implicit's views on this point have not been altogether consistent, I note that the goals and purposes of many of these prior art systems are similar or identical to the accused products, making modification of the Juniper system practical and cost effective. Moreover, many of these prior art systems are adaptable, further minimizing any cost impact associate with modification. Thus, to the extent Implicit contends that any of these many flow-based prior art references do not disclose each and every element of the patents-in-suit, they would constitute non-infringing alternatives.

299. As noted above, however, in view of Implicit's unclear and inconsistent positions, I reserve the right to provide additional details on this subject in response to Implicit's shifting positions, or should Implicit identify a coherent theory.

10 MISCELLANEOUS

300. I am being compensated at a rate of $350/hour. My compensation is not contingent in any way on the outcome of this case. I have not published any articles or papers during the last 10 years. A list of materials I considered in writing this report is contained in **Attachment A**. A list of cases in which I have testified as an expert at trial or at deposition is included herein by reference as **Attachment C**. My current CV is provided as **Attachment B**, and also included herein by reference.

301. The documents I have specifically referenced in this report and in **Attachment A** are exemplary and are intended to aid understanding. I may rely at trial on these, as well as other (1) documents and things produced in this action by the parties or third parties; (2) deposition exhibits;

(3) deposition testimony of the parties and third parties; (4) Implicit' responses to Juniper's discovery requests; (5) the documents referenced in Dr. Nettles's report and any other materials relied upon by any of Implicit's experts; (6) publicly available materials.

302. I may also rely on visual aids and demonstrative exhibits that may be prepared based on these materials that illustrate the bases of my opinions, including the materials mentioned in this report and in **Attachment A**. These visual aids and demonstrative exhibits may include, for example, claims charts, patent drawings, excerpts from patent specifications, file histories, deposition testimony, or diagrams or other graphical presentations describing the technology of the patents-in-suit.

303. I also expect to testify at trial with respect to the subject matter of my report and matters addressed by any expert testifying on behalf of Implicit if asked about these matters by the Court or by the parties' counsel. I may also testify on other matters relevant to this case if asked by the Court or by the parties' counsel.

304. To the extent that Implicit or its expert witnesses offer new opinions or evidence in this case, I reserve my right to supplement, amend, or modify the opinions set forth in this report.

Dated: September 11, 2012

Peter Alexander, Ph.D.

EXHIBIT B

PETER ALEXANDER, Ph.D.

501 ½ Larkspur Avenue
Corona del Mar, CA 92625
Office (949) 760 9990 Cell (949) 554 3934
email: peter.alexander@roadrunner.com

EDUCATION

Ph. D., Electrical Engineering, Massachusetts Institute of Technology, 1971
MS, Electrical Engineering, University of Illinois, 1967
BS, Electrical Engineering, University of Canterbury, New Zealand, 1965

PROFESSIONAL AFFILIATIONS & AWARDS

Fulbright Scholar, 1965
National Science Foundation – Small Business Innovation Research, 1988
Dept. of Energy – Small Business Innovation Research, 1988
Member Association of Computing Machinery (ACM)
Member IEEE Computer Society

PROFESSIONAL EXPERIENCE - SUMMARY

- Technical Expertise - Computer software design, development & deployment
- Forensic data acquisition and analysis
- Microsoft Visual Studio component and application design
- Web integration of authentication services, streaming media services, ad displays, content feeds
- Implementation of real-time and media streaming systems
- Architecture and design of complex business systems involving database back ends
- Oracle 8i, 9i, SQL Server 2000, and DB2 database technology.
- Java, C, C++, Visual Basic, assembly language programming
- Embedded microprocessor designs
- Network equipment design and manufacture (LAN cards, routers, bridges)
- Security, authentication, networking, firewalls, hacking countermeasures, backups, archives and service level agreements for customer-outsourced data.

Domain Expertise - Client-server and web-based software applications

- ERP systems – financial, distribution, manufacturing, SF automation applications (Platinum Software)
- eCommerce - Secure web transactions, authentication (InfrastructureWorld.com, Syntricity, Inc.)
- Semiconductor manufacturing – yield analysis, semiconductor defect analysis (Syntricity, Inc.)

PROFESSIONAL EXPERIENCE

2003 to Present **Independent Computer Consultant**

Independent computer technology consultant since February 2003, offering advisory services for information technology organizations, venture capital groups and the legal profession. Services have been provided for projects in the following areas:

- **Computer software development contracts for Internet and client**-server software implementation.
- Disaster recovery for large scale IT operations.
- Definition of product design and market positioning for a "distance learning" product designed to enable training of employees via a web server application.
- Forensic analysis of computer data and electronic discovery from computer disks.
- Definition of a web-based eCommerce system to provide secure business transactions. Developed architectural plans and database schema for Oracle database implementation.
- Provided technical consulting services regarding the behavior of certain scripts used for a mIRC chat server (Jedi 2.1). Analyzed the internal architecture of the Jedi relay chat system. At issue was the behavior of a Jedi server when a remote user (client) uploaded a file for distribution to other clients. Analyzed the upload scripts and formed a preliminary opinion regarding the functional behavior of the code.
- Provided research on spyware products that are downloaded to a user's client computer while browsing the web. Determined the behavior and installation mechanisms for this class of spyware products, and provided consulting on ways of removing them.
- Provide consulting services to eBusiness clients for the creation of SOAP integration of database services. Provided planning assistance for a large, ASP (Active Server Pages) web site to implement live business information feeds.
- Analyzed web application server prototypes from Apache Software Foundation reference implementations such as the Apache HTTP server project, Jakarta Tomcat Project (J2EE Servlet and JSP Web container), Turbine a servlet based application framework, and the Velocity Template Engine reference designs

2001 to 2003 **Syntricity, Inc., San Diego, CA**
Vice President Technical Operations

Syntricity is a supplier of application software to the semiconductor industry. Its customers include Intel, Sun Microsystems, Broadcom, Qualcomm and Conexant. Products are installed on Unix and NT 4.0 web servers at the customer site as Intranet solutions (Enterprise), or on Unix servers at the Syntricity data center, which offers an ASP style subscription services.

Developed a large capacity warehouse architecture and deployable warehouse solution to support defect, FBM, lot history, non-lot equipment, and other data storage requirements to support analysis of yield and production forecasts from test data acquired in the semiconductor manufacturing environment. This system was built on Oracle 8i and 9i

commercial RDBMS products. Implemented a comprehensive set of statistical analysis tools including multivariate regression and confidence level testing to facilitate yield trend and production scheduling for semiconductor manufacturers. Developed a messaging transaction system - "Integration Server" for WIP/MES back-end business processes. Various technologies were incorporated including: RMI, from a JMS input queue and JNDI for naming services.

The design was implemented with Oracle 9.2 loader and schema validation technology, and required user ETL data to be formatted as XML documents. Java 2 SE was the implementation platform language. The web server, based on the Tomcat open source code from the Apache Consortium, was enhanced to provide comprehensive access control according to user class, and included integrated end-user script-based customization (using the open source Python interpreter). XML objects were used extensively to represent web server data structures in the core implementation. High volume datalog insertion (ETL) back-end functionality was implemented for user uploads via FTP. An earlier generation C-coded CGI version was also supported. All products developed were web server solutions, with access via a standard browser. Responsible for creating and managing design teams as well as quality assurance, configuration management, technical hosting operations, and technical documentation groups. Responsible for product functional specifications, source code control, defect tracking, configuration/build management, and application validation.

Responsible for a team of 40 people across four groups that included software engineering, database design, quality assurance and technical operations. Java 2 was the implementation platform language, and Oracle 8.1.6 was used for the warehouse database. All products were designed as web server solutions, with access via a standard browser. Responsible for creating and managing design teams as well as quality assurance, configuration management, technical hosting operations and technical documentation groups. Detailed understanding of source code control, defect tracking, configuration management and build tracking.

Customers using the hosting subscription center run under contractual Service Level Agreements (SLA). The ASP hosted service is currently implemented on an E4800 Sun application server running Solaris 8, connected to an Oracle database server (Oracle 8.1.6). Storage totaling approximately 1 Terabyte is provided through a combination of the EMC Clarion System 1 storage arrays (Raid-5) accessed via fiber channel, and network attached storage using the Network Appliances NetApp devices. Veritas SANPoint Foundation Suite HA for Solaris is used as the storage management tool.

2000 to 2001 **InfrastructureWorld, San Francisco, CA**
Chief Technology Officer

Infrastructureworld, a spin off from Bechtel Enterprises, offered services through a collaborative web site for large-scale construction projects. Managed development staff of 10, and operational staff of 3 to create, enhance and maintain the live web site. Responsible for a new web server implementation based on NT4.0 and Windows 2000 technologies, to support authentication through certificates, user access control via authenticated account login, SSL extranet connections and document encryption.

Investigated Public Key/Private Key encryption authentication mechanisms before selecting Windows NT integrated challenge/response authentication as the preferred authentication technique.

Each business client was hosted as a separate virtual web site with secure access to content that described the client's projects offered for bid. Functionality included insurance and financing RFP's, document management, and project collaboration. In addition, each web site offered integration of multimedia content for promotion of client projects, including steaming video and audio content. Implemented a secure data access system using native NT operating system authentication services. All documents and files were transmitted via 128-BIT SSL using server side certificates for server authentication to the client browser. Automatic virus scanning and cleaning was implemented for all documents and files uploaded by users to the web server. The operational web server site was implemented with a two-tier server configuration using Raid (redundant) storage.

1999 to 2000 **CareerPath.com, Los Angeles, CA**
Senior Vice President, Technology

Management of Operations and Development teams. Lead the company's Web site re-architecture project, providing higher levels of Web server and Oracle database performance. Implementation of methodologies for project management, code review, quality assurance, and defect tracking. Managed the operations group (40) supporting the production Web site, encompassing wide area networking, Unix administration, Oracle DBA support, HTML authoring and quality assurance teams. Managed the software development staff (25) which created new technology infrastructure and dynamic page content using Java middle tier servlets, Java Beans, JSP presentation components, and Oracle technology. Object oriented programming techniques were applied through use case analysis.

Created a feed management system, written using server-side Java parsing technology, to processes Web job postings harvested from Web spider technology. Later enhancements included development of a Content Management system (using XML page representation), vertical affiliate co-branding system, transparent registration and login across a federation of partnership Web sites, and a comprehensive on-line reports server.

1999 **Charles Schwab Online Trading, Phoenix, AZ**
Independent Consultant

This project involved disaster recovery cold site planning for the Charles Schwab Online Web Trading facility. The web capability was capable of handling 200,000+ concurrent users, and was implemented with 300 load-sharing IBM gateway servers working behind eight Cisco Catalyst 7500 routers. The traffic was distributed across the front line servers and routed to an ensemble of 200 middle tier servers, which manage the business objects and execute the trades. In the third tier, customer financial and demographic data was maintained on a group of seven IBM mainframes running DB2. A design was formulated that gives the Schwab organization a contingency backup system in the event of catastrophic failure of the main site.

1997 to 1999 **Platinum Software Corp., Irvine, CA**
(Re-named as Epicor Software Corp)
Vice President, Development

Reported to the President until 7/1/98, then reporting to the Executive VP of Product/Marketing. Member of the Executive Committee (top 8 executives in Platinum).

Managed a team of about 100 contributors working on Windows NT-based Client-Server systems. The Department was organized into five functional groups each headed by a Director-level manager, and includes ERP application development, technology/tools development, Windows/DOS legacy systems, QA and documentation teams.

This scope of the development effort encompassed the Platinum ERP client-server product suites, which include financial and distribution applications. These applications, based on Microsoft SQL Server technology, are implemented within a two-tier tool set, and involved 500 tables and more than 2000 stored procedures. Responsibilities also included all ERP integration tools and application content. The ERP integration suite allows remote transaction integration of customer relationship management, sales force automation, distribution, manufacturing, and financial applications, as well as OLAP business intelligence reporting via client-side components and Microsoft DSS.

Direct management of teams deploying MS Message Queue, MS Transaction Server, MS SQL Server 6.5 and 7.0, NT 4.0, XML, and business object technology. Successfully launched a high volume test group to establish performance of the client-server products under stress, and to determine their scalability. Built an architectural team to design and implement 3 tier, thin-client framework, using Java business objects running on an application server. Established effective methodologies for fostering cooperation in development projects requiring the participation of geographically remote design and development teams.

Developed a pure Java client-server system to support an end user form builder application. The objective of this project was to demonstrate an application server-centric 3-tier architecture. Application servers are dispensers of services, and a Java compilation service was the central technology being demonstrated. The pilot development showed building block to support a broad variety business object models, forms architectures, and rules engines. (JDK 1.1)

A full-functionality GUI in Java was also developed, along with a Java-based Customization Workbench, built, using the form class metaphor, which is familiar to users of Visual Café, JBuilder, or J++ 6.0. The Java Swing classes were used in the implementation.

Created a prototype OLAP decision support analysis system. The Platinum Info Report pack contained OLAP cubes specific to the Platinum ERA financial, distribution and manufacturing data. OLAP cubes were implemented on top of core Platinum applications to enable multi-dimensional analysis on key business drivers such as sales activity. Reports contained in the Info Report Pack were designed to leverage graphical,

WYSIWYG reporting and analysis capabilities of Crystal Info, a third party reporting package, including: presentation-quality formatting, charts, graphs, drill-down, top "N" analysis, search, sort and criteria selection.

1994 to 1996 **Quixote Corp., Chicago, IL**
Vice President, Technology (Reporting to the President of Legal Technologies, Inc.)

Legal Technologies, a subsidiary of the Quixote Corp., was formed as a consortium of four companies operating in the legal vertical market to address law office automation, and related legal services.

Formulated opinions and strategies for technology-related products being considered for acquisition or development by the company. Assisted the corporate legal counsel in the interpretation of patent claims for the purpose of defending or initiating lawsuits. Participated in negotiations with plaintiffs to bring about resolution of legal conflicts.

Managed teams developing Windows client-server database applications for the legal/judicial markets. These products involved MFC C++ and Visual Basic 4.0, NT/SQL Server and Btrieve technology.
Development of video deposition applications. Encoding of VHS tapes from video depositions into MPEG using Real Magic encoders. Annotation of video to allow rapid search of content and synchronization of video to the written transcript. The user was provided with the ability to jump to a specific page and line in the written transcript with automated tracking to the correct location in the displayed video. Similarly, the rolling video stream would automatically scroll the text display.

1989 to 1992 **Fibronics International, Lowell, MA**
General Manager, Spartacus subsidiary

Successfully developed software products for TCP/IP and Unix-related networking applications. These products performed routing and bridging functions for Internet packets, at data rates of up to 100Mbps across FDDI fiber networks. Technology involved embedded C-coded Intel RISC CPU's and Motorola 68K series microprocessors. In addition, a suite of Internet-protocol software packages was developed for integration of mainframes with other Fibronics TCP/IP LAN and WAN products. Networking software developed for mainframe computers included: TCP/IP protocol stack; Network File System (server) package; File Transfer Protocol (FTP) server module; Simple Mail Transfer Protocol (SMTP); 3270 terminal emulators; X-windows client functionality.

The NFS application was the result of a nine month development effort involving a team of six software developers. The complete design, including all architectural components, was formulated from Sun Microsystems documentation, and the resulting concept was implemented from scratch in C++. A parallel research study of the competing Andrew File System was also carried out. The NFS product was deployed by several large Fibronics customers.

1982 to 1988 **Numerix Corporation, Newton, MA**
President/CEO

Founder and CEO of a high-technology computer company with 130 employees. Led the company from its startup in May 1982 through rapid growth. Revenues grew from $0 to $10M in three years.

Supervised a software and hardware development team of 50 people, and a total employee headcount of 130. Lead contract negotiations with vendors, partners and investors. Intimately involved with product engineering, quality assurance and field reliability problems.

1975 to 1981 **CNR, Inc., Newton, MA**
Vice President

Managed software and hardware design teams for defense-related projects. These included: real-time data acquisition using microprocessor systems, real-time communications channel simulators, embedded Intel and Motorola microprocessor systems, designs for the application of the Global Positioning System to air traffic control, synchronization of the DOD world-wide communication system using atomic clocks to facilitate the distribution of a high-precision time reference.

1972 to 1975 **Univ. of Auckland - New Zealand**
Asst. Prof., EE

Performed teaching and graduate research responsibilities for electrical engineering and computer science. Taught graduate level courses on control systems, communication systems, microprocessors and integrated circuit design. Taught undergraduate courses involving electromagnetic theory, electronic circuit design and mathematics